Software libre



José María Barceló Ordinas
Jordi Íñigo Griera
Ramón Martí Escalé
Enric Peig Olivé
Xavier Perramon Tornil

XP06/M2105/01496

Redes de computadores



David Megías Jiménez

Coordinador

Ingeniero en Informática por la UAB.

Magíster en Técnicas Avanzadas de Automatización de Procesos por la UAB.

Doctor en Informática por la UAB.

Profesor de los Estudios de Informática y Multimedia de la UOC.

Jordi Mas i Hernández

Coordinador

Ingeniero de software en la empresa de código abierto Ximian, donde trabaja en la implementación del proyecto libre Mono. Como voluntario, colabora en el desarrollo del procesador de textos Abiword y en la ingeniería de las versiones en catalán del proyecto Mozilla y Gnome. Es también coordinador general de Softcatalà. Como consultor ha trabajado para empresas como Menta, Telépolis, Vodafone, Lotus, eresMas, Amena y Terra España.

Enric Peig Olivé

Coordinador

Doctor ingeniero de Telecomunicaciones por la Universitat Pompeu Fabra. Actualmente trabaja en la especificación de metadatos aplicados al comercio electrónico. Es profesor en los Estudios de Informática de la UPF.

José María Barceló Ordinas

Autor

Doctor ingeniero de Telecomunicaciones por la Universidad Politécnica de Cataluña. Actualmente trabaja en la evaluación de redes ATM y en la modelización de tráfico en redes informáticas. Es profesor del Grupo de Redes de Computadores en la Facultad de Informática de Barcelona.

Jordi Íñigo Griera

Autor

Ingeniero de Telecomunicación por la Universitat Politècnica de Catalunya. Actualmente es Director de Desarrollo de Software de Safelayer Secure Communications, S.A. Ha sido Director Técnico del esCERT (Equipo d e Seguridad para la Coordinación de Emergencias en Redes Telemáticas) de la UPC. Es profesor del Grupo de Redes de Computadores en la Facultad de Informática de Barcelona.

Ramon Martí Escalé

Autor

Doctor ingeniero de Telecomunicación por la Universitat Politècnica de Catalunya. Actualmente trabaja en la seguridad en aplicaciones distribuidas de comercio electrónico de información multimedia. Es profesor de los Estudios de Ingeniería de Telecomunicación de la Universitat Pompeu Fabra de Barcelona.

Xavier Perramon Tornil

Autor

Doctor ingeniero de Telecomunicación por la Universitat Politècnica de Catalunya. Actualmente trabaja en el diseño y estandarización de sistemas de documentación multimedia. Es profesor de los Estudios de Informática de la Universitat Pompeu Fabra de Barcelona.

Primera edición: marzo 2004

Av. Tibidabo, 39-43, 08035 Barcelona
Material realizado por Eureca Media, SL

Depósito legal: B-7.598-2004

Índice

Agradecimientos

Los autores agradecen a la Fundación para la Universitat Oberta de Catalunya (http://www.uoc.edu) la financiación de la primera edición de esta obra, enmarcada en el Máster Internacional en Software Libre ofrecido por la citada institución.

ANOTACIONES

Introducción

Las redes de ordenadores actuales son una amalgama de dispositivos, técnicas y sistemas de comunicación que han ido apareciendo desde finales del siglo XIX o, lo que es lo mismo, desde la invención del teléfono. El teléfono, que se desarrolló exclusivamente para transmitir voz, hoy se utiliza, en muchos casos, para conectar ordenadores entre sí. Desde entonces han aparecido las redes locales, las conexiones de datos a larga distancia con enlaces transoceánicos o satélites, la telefonía móvil, etc. Mención especial merece la red Internet dentro de este mundo de las comunicaciones a distancia. Nadie duda de que hoy en día constituye una red básica de comunicación entre los humanos.

Este curso ofrece una visión de las redes informáticas en general y de la red Internet en particular.

En la primera parte, introduciremos las ideas y los conceptos básicos de las redes de ordenadores. Siguiendo un hilo histórico, presentaremos los diferentes mecanismos que se han utilizado y se utilizan para comunicarse a distancia. Presentaremos igualmente el concepto de arquitectura de protocolos, fundamental en sistemas distribuidos, y el modelo de referencia OSI como un ejemplo paradigmático de ello. Aunque hoy en día este modelo no disfruta de una gran popularidad, sus virtudes pedagógicas están más que demostradas: a partir de él es fácil estudiar y entender otras arquitecturas, como la arquitectura Internet en torno a la cual gira todo el curso.

La segunda parte está dedicada al estudio de las redes de área local. Presentamos de forma descriptiva los diferentes tipos de redes que existen, las ideas básicas de su funcionamiento y la noción de cableado estructurado, clave en el gran auge que han tenido últimamente las redes de área local.

En la tercera parte se verán los fundamentos de la red Internet. Lo que se conoce como *red Internet* es un conjunto heterogéneo de redes interconectadas. Precisamente, es la capacidad de homogenei-

Nota

Internet es un apócope de *internetworking* (interconectando redes).

zar lo que de hecho es heterogéneo, lo que ha catapultado la red Internet a su estatus actual.

Los protocolos que distinguen la red Internet como una unidad son el IP (*Internet protocol*) y el TCP (*Transmission control protocol*). Estos protocolos no son los únicos, pero sí los más importantes de entre los que se necesitan para hacer funcionar la red Internet. Por este motivo, a todos en conjunto se les llama normalmente *pila TCP/*IP (*TCP/IP stack*).

En concreto, en esta parte se describe el protocolo IP y sus más inmediatos colaboradores (ARP y ICMP), así como los mecanismos de acceso a Internet de que disponemos: a través de una red de área local o un enlace telefónico, ya sea mediante PPP y un módem tradicional o, más recientemente, mediante ADSL.

TCP/IP no es un estándar *de iure*. Ningún organismo internacional de estandarización se ha encargado de emitirlo. Por el contrario, el funcionamiento de sus protocolos está recogido en unos documentos llamados RFC (*request for comments*), que son propuestas que se han hecho sobre el funcionamiento de un protocolo concreto, o de una parte. El proceso es simple: una vez hecha pública una propuesta, si nadie pone ninguna objección, ya se considera aprobada y lista para ser implementada.

Nota

Las RFC se pueden consultar en la siguiente dirección: http://www.ietf.org.

Además de consultar este material didáctico y la bibliografía recomendada, en que se explican los protocolos de una forma pedagógica, se recomienda leer alguna RFC, aunque sólo sea para hacerse una idea del proceso que ha seguido la Red desde sus inicios

En la cuarta parte, describiremos los protocolos de aplicación más utilizados actualmente en Internet y los programas más habituales que los implementan, como son la conexión remota (telnet, rlogin), la transferencia de archivos (FTP), el correo electrónico (SMTP, POP, IMAP), las news (NNTP), el WWW (HTTP) y la mensajería instantánea.

Todos estos programas se conocen como *aplicaciones distribuidas*, puesto que están formadas por distintas partes que pueden estar ejecutándose en máquinas diferentes. Esta dispersión de partes de programas obliga a definir una manera de dialogar entre ellas.

Veremos pues, antes de empezar la descripción de las diferentes aplicaciones, este concepto de programación distribuida y el modelo cliente/servidor que es el que sigue mayoritariamente.

Las aplicaciones Internet permiten conocer las máquinas y los servicios a través de nombres, y no con números que es como trabajan IP, TCP y UDP. Alguien tiene que encargarse de la asociación de los nombres con las direcciones numéricas y este alguien es el servicio DNS (*Domain Name System*). También trateremos este tema antes de describir las aplicaciones.

ANOTACIONES

Objetivos

Con los materiales de este curso se pretende que el lector alcance los objetivos siguientes:

1. Conocer las diferentes tecnologías que se utilizan en la actualidad para transmitir información a distancia, y comprender cuándo y por qué aparecieron.
2. Conocer el modelo de referencia OSI, sus utilidades y sus limitaciones, y ser capaz de entender la motivación de cada uno de sus niveles.
3. Conocer los principios básicos de funcionamiento de las redes de área local tanto cableadas como inalámbricas, las topologías posibles y las diferentes políticas de acceso al medio.
4. Conocer el concepto de cableado estructurado, entender el papel que en él juegan los concentradores y saber diferenciar topología física y topología lógica.
5. Entender los principios de funcionamiento del protocolo de nivel de red IP: la asignación de direcciones y el direccionamiento.
6. Aprender el funcionamiento de las redes de acceso a Internet más comunes: acceso LAN y acceso per red telefónica mediante PP o ADSL.
7. Entender el funcionamiento de los protocolos de transporte y saber en qué principios se basan.
8. Conocer algunas utilidades de uso común que permiten descubrir algunas interioridades de estos protocolos de red y transporte.
9. Comprender el modelo cliente/servidor, que sirve como base de la implementación de aplicaciones distribuidas y el modelo *peer-to-peer*, complementario del anterior.
10. Comprender el funcionamiento del DNS, el servicio de nombres de dominio, que da soporte al resto de aplicaciones.

ANOTACIONES

11. Conocer las aplicaciones telnet y rlogin, que proporcionan el servicio de conexión remota a otros ordenadores (principalmente en el entorno GNU/Linux), y las aplicaciones que proporcionan en Internet los servicios de transferencia de archivos, correo electrónico, *news*, WWW y mensajería instantánea, y sobre todo los protocolos que siguen.

ANOTACIONES

I. Introducción a las redes de computadores

1. Breve historia de las comunicaciones

Desde que el ser humano tiene capacidad de comunicarse ha desarrollado mecanismos y sistemas que les permiten establecer esta comunicación a distancias superiores de las alcanzadas por sus propios medios.

Al poco de aparecer los ordenadores, se sintió la necesidad de interconectarlos para que se pudiesen comunicar entre sí como lo hacemos los humanos.

En esta unidad nos planteamos repasar la historia de estos sistemas de comunicación, pensados para ser usados por los humanos y que, después, han ido evolucionando para interconectar ordenadores.

Fijamos el inicio de este recorrido histórico en el teléfono. El teléfono no fue el primer sistema de telecomunicación, pero sí el más antiguo de los que hoy en día se utilizan habitualmente. Mucho antes se habían utilizado sistemas ópticos que, con la luz del sol y juegos de espejos, permitían comunicarse desde distancias considerables. Con posterioridad, a mediados del siglo XIX, se inventó el telégrafo. Estos sistemas, sin embargo, han caído en desuso (excepto usos marginales), mientras que la red telefónica se mantiene como un sistema de comunicación de primer orden.

1.1. El teléfono

En 1878, Alexander Graham Bell mostró su "máquina eléctrica parlante" y cómo podía mantener una conversación a distancia entre dos de estos aparatos unidos por un hilo eléctrico.

Nota

Recientes investigaciones han hecho salir a la luz una historia curiosa: parece claro que el inventor del telé-

Nota

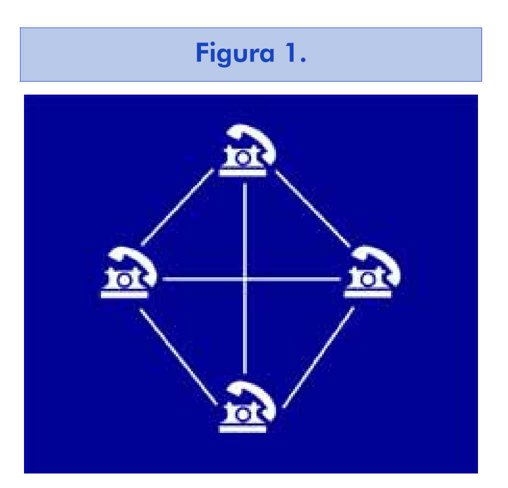
Podéis encontrar la historia completa de este episodio en la siguiente dirección: http://www.popular-science.net/history/meucci_bell.html.

fono fue un italiano llamado Antonio Meucci, pero no patentó su invento porque no tenía suficiente dinero para hacerlo. Bell se apropió del invento y lo patentó.

Al principio, los pocos teléfonos que existían se utilizaban en entornos cerrados, particulares. Servían para interconectar dos espacios. A medida que el número de teléfonos instalados crecía, el interés por mantener múltiples comunicaciones también lo hacía: era preciso pensar en la manera de interconectarlos. Nacía la idea de red de comunicaciones.

Una posible manera, bastante inmediata, de interconectar todos los aparatos sería lo que se puede observar en la figura siguiente:

Figura 1.

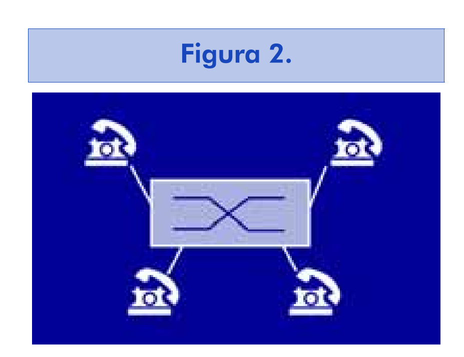


Es evidente que este modelo de conexión, "todos con todos", es completamente inviable: para cada aparato nuevo que se incorpora a la red, se precisa un gran número de conexiones nuevas. Para hacernos una idea, una red "todos con todos" de cincuenta teléfonos necesita 1.225 líneas de conexión y, en cada teléfono, un dispositivo que permita cuarenta y nueve conexiones.

Para solucionar este problema, aparecieron compañías que ofrecían un servicio de **commutación**: hacían llegar un cable hasta cada teléfono y conectaban los cables de los teléfonos que deseaban establecer una comunicación. De este modo, cada aparato disponía de una sola conexión y no era necesario establecer ninguna variación en la misma para incorporar nuevos aparatos a la red.

ANOTACIONES

Figura 2.



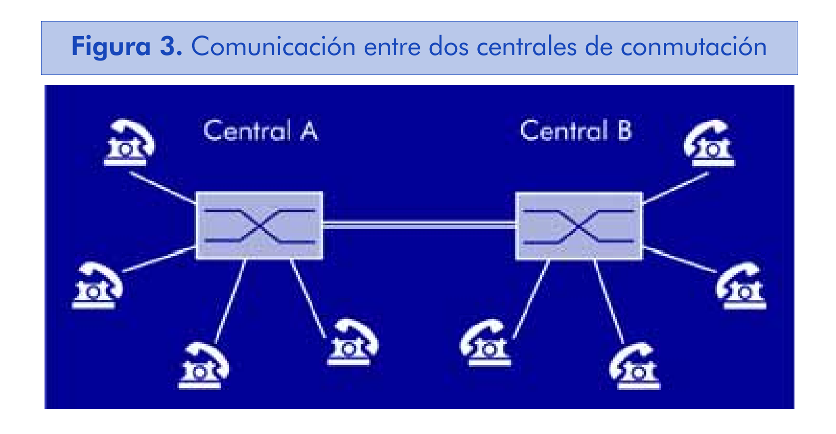
De aquí provienen términos hoy tan comunes como *abonado* (el usuario que se abona a una central), *bucle de* ***abonado*** (el cable que une al abonado con la central) o *central de* ***conmutación***.

La tarea de conmutar las conexiones, al principio, se hacía a mano. Cuando alguien quería realizar una llamada, descolgaba y pedía a la operadora que le conectara con quien deseaba hablar. Una vez finalizada la comunicación, la operadora desconectaba los cables y, así, las líneas quedaban preparadas para recibir otras llamadas.

Las operadoras humanas fueron sustituidas progresivamente por ingenios electromecánicos: las **centralitas**. Se incorporó a los teléfonos un disco con números para "marcar" el número del destinatario de la llamada. La centralita descodificaba este número para saber entre qué dos cables era preciso establecer la comunicación.

Este servicio de conmutación empezó en el ámbito local: un barrio, un pueblo, una ciudad. El paso siguiente consistió en ofrecer conexiones a larga distancia, conectando centrales locales entre sí directamente, o por medio de **centrales de tráfico**.

Figura 3. Comunicación entre dos centrales de conmutación



Entre las dos centrales locales se establece un enlace con diferentes cables independientes, de manera que los abonados de una de éstas



pueden, además de conectarse entre ellos, conectar con los abonados de la otra: se elige un cable de los que forman el enlace, se conecta con el abonado local y se pide a la otra central que conecte el enlace con el abonado destino, si no está ocupado con otra llamada.

La conexión entre las dos centrales comporta un primer escollo importante: es preciso decidir con cuántas líneas diferentes se llevará a cabo.

Supongamos que la central A de la figura anterior proporciona servicio a cien abonados y la B, a doscientos cincuenta. Parece que, si se pretende dar el mejor servicio posible, se necesitan cien líneas para que todos los abonados de la central A puedan hablar de manera simultánea con otros tantos de la central B.

No obstante, la probabilidad de que todos los abonados de una central realicen una llamada al mismo momento es muy baja, puesto que las llamadas telefónicas son, en general, cortas y esporádicas. Por tanto, es completamente innecesario que la conexión entre las dos centrales contemple todas las llamadas posibles: esta situación no se dará nunca y tiene un coste exagerado.

Unos modelos matemáticos bastante complejos permiten calcular el número concreto de enlaces que se precisan a partir de la estadística de las llamadas que sirven las centrales (la frecuencia de aparición y su duración).

Nota

A.K. Erlang, ingeniero danés de principios del siglo xx, estableció los modelos matemáticos que se utilizan para medir el tráfico telefónico.

Se puede encontrar mucha información al respecto en la dirección siguiente:

http://www.erlang.com

Supongamos que en el ejemplo anterior estos modelos nos dan veinticinco enlaces. Si en un momento dado hay veinticinco llamadas en curso entre A y B y llega otra llamada, no tendrá ningún camino disponible y, por consiguiente, no se podrá establecer. Esta situación se denomina *bloqueo*: el abonado a quien se quiere llamar no está ocupado; sin embargo, no se puede encontrar un camino libre por la red para establecer la comunicación.

De esta situación se desprenden dos ideas fundamentales en relación con la red telefónica:

- La conmutación de circuitos requiere pasar por tres fases para cada comunicación:

– **Establecimiento de llamada.** Cuando se solicita iniciar una conversación, es preciso averiguar si el destinatario está disponible y,

ANOTACIONES

en caso afirmativo, debe buscarse un camino libre en la red, que incluye conmutadores dentro de las centrales y enlaces entre las mismas.

– **Comunicación.** Una vez establecido el circuito, los interlocutores se intercambian información.

– **Liberación de recursos.** Acabada la comunicación, se liberan los recursos utilizados (enlaces entre centrales y conmutadores dentro de las centrales).

• El hecho de que los recursos estén ocupados en exclusiva mientras dura la comunicación hace que las compañías que ofrecen el servicio cobren según la duración de la llamada: se penaliza el uso extensivo de los recursos. De este modo, el usuario se apresura en acabar la comunicación y dejar los enlaces libres, disminuyendo así la probabilidad de bloqueo.



La red telefónica constituye una red de conmutación de circuitos. Para llevar a cabo una comunicación, es preciso establecer un circuito entre los dos extremos por medio de la red. Mientras dura la comunicación, se ocupan unos recursos en exclusiva, aunque no haya intercambio de información. Las compañías cobran el uso de los recursos por tiempo de ocupación.

Pronto, el sistema telefónico pasó a ser una cuestión nacional. Los estados desarrollaban sus redes según sus criterios y gustos. Se creó un organismo, el CCITT (Comité Consultivo Internacional de Telegrafía y Telefonía, Comité Consultatif International Télégraphique et Téléphonique), para armonizar los sistemas nacionales y permitir las comunicaciones entre países mediante centrales de tráfico internacionales.

Nota

El CCITT es un organismo internacional patrocinado por las operadoras de telefonía, dedicado a tareas de normalización en el ámbito de las telecomunicaciones. El 1 de marzo de 1993 pasó a llamarse ITU-T (International Telecommunication Union Standardisation Sector).

Hemos comentado que entre las centrales existe una serie de líneas que permiten la conexión entre abonados de diferentes centrales. Al principio era realmente así: si se decidía que entre dos centrales era preciso disponer de cincuenta enlaces, se ponían cincuenta cables entre ellas. Sin embargo, con el progresivo aumento

ANOTACIONES

Nota

Multiplexar significa hacer pasar diferentes comunicaciones independientes por el mismo medio de transmisión.

de enlaces necesarios, este sistema pronto fue totalmente inviable y fue preciso recurrir a una técnica ya conocida en radiodifusión: la multiplexación.

La técnica de multiplexación que se aplicó a la telefonía fue la multiplexación en frecuencia: se modulan los diferentes canales de entrada a distintas frecuencias portadoras, de manera que puedan viajar por el mismo medio sin interferirse. Se aplican filtros a la recepción que permiten separar los distintos canales multiplexados.

Ejemplo

Hacemos lo mismo al escuchar la radio o al ver la televisión. Hasta nuestra antena llegan todos los canales emitidos; con el dial y el selector de canales, respectivamente, seleccionamos el canal (la gama de frecuencias) correspondiente a la emisora que queremos recibir. Es decir, el dial o el selector de canales de la televisión constituyen los filtros que separan, en la recepción, los diferentes canales multiplexados.

El número de canales diferentes que pueden viajar por un medio multiplexado depende del ancho de banda de la señal y de la capacidad del medio.

Por lo que respecta a la capacidad del medio, no posee la misma un par de hilos que un cable coaxial o que una fibra óptica.

En cuanto al ancho de banda, en el caso de la voz, debería ser de 19.980 Hz (que es un ancho de banda considerable) puesto que el oído humano es capaz de distinguir frecuencias entre los 20 Hz y los 20.000 Hz. No obstante, a raíz de estudios que se llevaron a cabo sobre las características de la voz humana, se llegó a la conclusión de que con mucho menos bastaba, puesto que la inteligibilidad de la voz se concentra en una banda bastante estrecha, entre los 300 Hz y los 3.400 Hz.

A partir de esta conclusión, se tomó una decisión que, a la larga, ha condicionado mucho el uso de la red telefónica: hacer el canal de voz de 4 kHz (entre 300 Hz y 3.400 Hz, más unas bandas laterales de guardia).

ANOTACIONES

Nota

Haber reducido el canal de voz a 4 kHz explica por qué se escucha tan mal la música por el teléfono: no hay ni graves ni agudos, sólo hay las frecuencias del medio.

A partir de aquí, se estandarizaron los diferentes niveles de multiplexación. El nivel básico es la agrupación de distintos canales de 4 kHz, el siguiente es una agrupación de multiplexados básicos, etc.

Nota

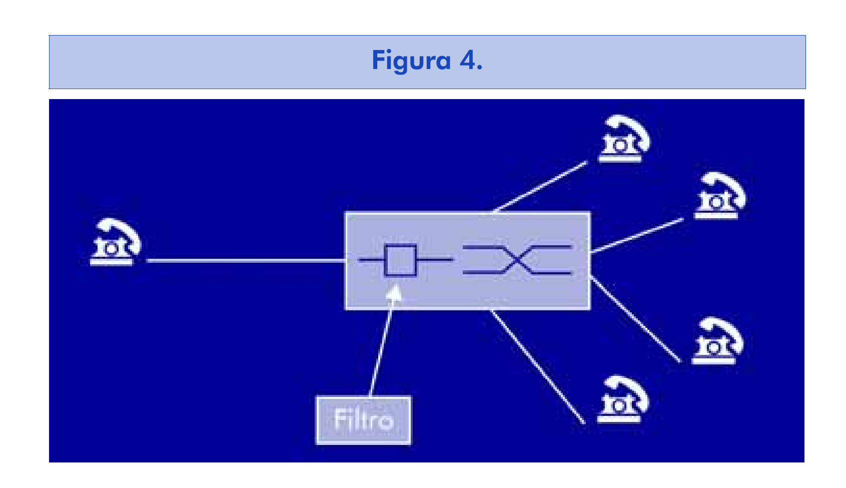
La jerarquía que propuso la compañía americana AT&T, y que ha acabado estandarizándose, es la siguiente:

Tabla 1.

Nombre	Rango	Ancho de banda	Canales de voz
Group	60-108 kHz	48 kHz	12
Supergroup	312-552 kHz	240 kHz	60
Mastergroup	564-3.084 kHz	2,52 MHz	600
Jumbogroup	0,5-17,5 MHz	17 MHz	3.600

A la entrada de la central local se encuentra un filtro que elimina cualquier frecuencia por encima de los 4 kHz. La señal de salida de este último es la que se multiplexa, conmuta y lleva hasta el destinatario.

Figura 4.



ANOTACIONES

Con todo ello, ya podemos dibujar un panorama completo de la red telefónica, tal como era hasta los años setenta:



La red telefónica es analógica, ubicua, trabaja con la técnica de conmutación de circuitos, con tarifación por tiempo de ocupación, con enlaces multiplexados en frecuencia y con canales limitados a 4 kHz.

1.2. Aparecen los primeros ordenadores

La década de los sesenta vio la aparición de los primeros ordenadores comerciales. Eran grandes, caros y poco potentes. Sólo organismos oficiales, grandes empresas o universidades podían comprarlo, y lo que es más normal es que sólo compraran uno (o algunos, pero no uno para cada usuario, como hoy día estamos acostumbrados a ver).

Nota

Al decir que eran máquinas poco potentes, evidentemente, es comparádolos con los actuales. Para la época, eran unas máquinas fantásticas.

Por ello, estos ordenadores llevaban sistemas operativos multitarea y multiusuario, para que diferentes usuarios, realizando distintos trabajos, pudieran utilizarlos simultáneamente. El acceso a dichos ordenadores se llevaba a cabo por medio de terminales sin ninguna capacidad de proceso, pasivos:

Nota

A los terminales pasivos, que coloquialmente se llaman *terminales tontos,* en inglés se les conoce como *dumb terminal* ('terminal mudo').

Figura 5.

Terminal pasivo
Ordenador central
Terminal pasivo
Instrucciones
Instrucciones
Resultados
Resultados

1.2.1. Los módems

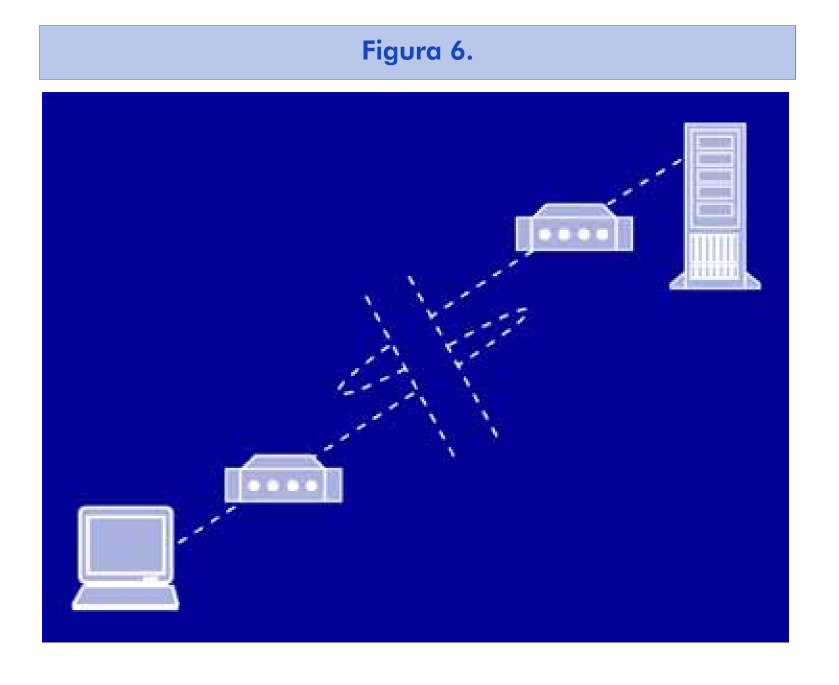
No tardó mucho en aparecer la necesidad de poder alejar los terminales de la unidad central para conectarse, por ejemplo, desde casa o desde una delegación al ordenador central.

Para poder realizar este acceso remoto, la primera solución que aportaron los ingenieros informáticos de la época fue utilizar la red telefónica que, por su ubicuidad, les ahorraba generar infraestructuras nuevas. Sólo se precisaba un aparato que adaptara los bits a la red (recordad que la red telefónica sólo deja pasar sonidos entre unos márgenes de frecuencia). Estos aparatos son los módems.

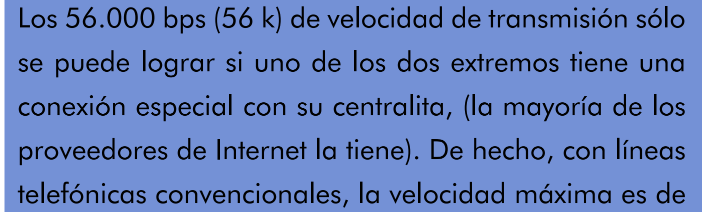
Nota

Módem es un acrónimo de *modulator-demodulator*, que se refiere a su función: modular (generar señales audibles según los valores de los bits) y demodular (generar bits a partir de las señales que recibe de la red telefónica).

Figura 6.



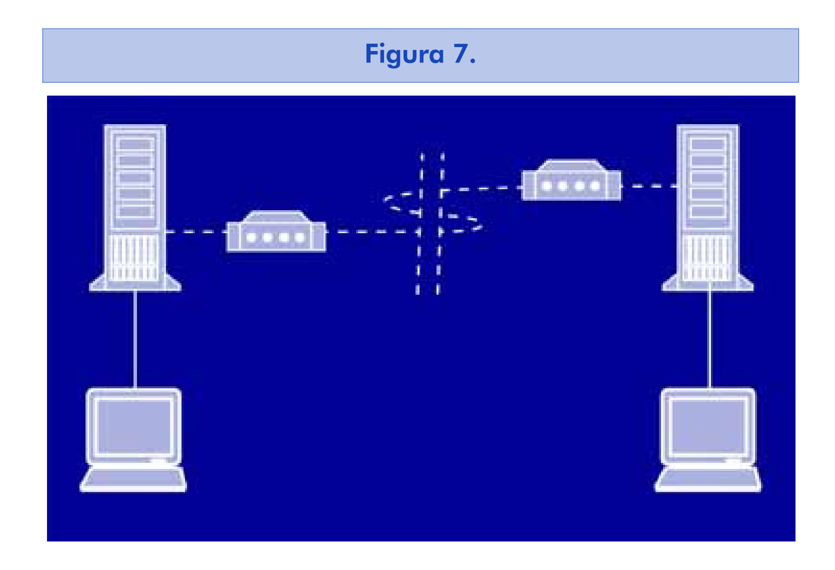
Los primeros módems eran de 300 bps y generaban dos tonos diferentes: uno para el 1 lógico y otro para el 0. En la actualidad, van a 56.000 bps, que es el máximo que permite la red telefónica convencional actual.



Los 56.000 bps (56 k) de velocidad de transmisión sólo se puede lograr si uno de los dos extremos tiene una conexión especial con su centralita, (la mayoría de los proveedores de Internet la tiene). De hecho, con líneas telefónicas convencionales, la velocidad máxima es de 33.600 bps.

Los módems no sólo servían para poder alejar los terminales pasivos de los ordenadores centrales, también permitían interconectar ordenadores entre sí.

Figura 7.



¡Esto ya es una red de computadores!

La tecnología de conmutación de circuitos se desarrolló en un origen para las comunicaciones telefónicas y una de sus características fundamentales era la ocupación en exclusiva de los recursos mientras duraba la conexión, lo que (como ya hemos visto) justificaba la tarifación por tiempo. Sin embargo, las comunicaciones informáticas no son cortas, intensas y esporádicas como las de voz. Al conectar un terminal a un ordenador central por medio de dos módems, no están pasando datos todo el tiempo que dura la conexión: puede haber largos periodos de tiempo en los que no pase ningún bit y momentos en los que haya un intercambio de datos intenso, aunque a una velocidad de transmisión mucho más baja que la que se puede mantener entre el terminal y el ordenador conectados directamente. Las facturas telefónicas empezaron a ser astronómicas, y desproporcionadas, respecto del uso real de la red.

1.2.2. Las redes de datos

Pronto las grandes empresas presionaron a las compañías telefónicas del momento para que desarrollaran redes pensadas para transportar datos, cuyo sistema de tarifación se ajustara al tráfico de datos

ANOTACIONES

real y permitiera más velocidad que los escasos 300 o 1.200 bps que se lograban utilizando la red telefónica. La respuesta fueron las **redes de conmutación de paquetes**.

El envío de datos no necesariamente debe llevarse a cabo en tiempo real (las transmisiones de voz, sí). Por tanto, no es preciso establecer el camino entre los dos puntos antes de empezar la transmisión y mantenerlo mientras dura el intercambio de datos. En lugar de ello, se empaquetan los bits que deben transmitirse y se dan a la central más próxima para que los envíe cuando pueda a la siguiente, y así sucesivamente hasta que lleguen al destino. Si cuando un paquete llega a una central todos los enlaces con la siguiente están ocupados, no pasa nada, lo hace esperar poniéndolo en una cola para enviarlo cuando haya un enlace disponible.



La transmisión por paquetes tiene la ventaja de que sólo ocupa los recursos cuando en realidad se utilizan, no siempre. Sin embargo, como contrapartida, es preciso soportar el retardo que pueda producirse entre que los paquetes salen del origen y llegan a su destino, que es variable, puesto que las esperas en las colas son aleatorias, dependen del estado de la red. Pero, como hemos dicho, en comunicación de datos este retardo es hasta cierto punto tolerable. Por lo que respecta a la cuestión económica, no tiene sentido que se cobre por tiempo de conexión: en las redes de datos se paga por bits transmitidos.

Existe otro peligro: los paquetes pueden perderse. Conviene tener presente que las colas son limitadas y, si llega un paquete cuando una ya está llena, no se podrá guardar y se perderá. Es preciso prever mecanismos que eviten dichas pérdidas y regulen el flujo de información entre los nodos de conmutación.

Nota

En España, la red de datos se llamaba *Iberpac*.

En la actualidad, para comunicaciones de datos se utiliza Frame Relay, la evolución natural de X.25.

Las compañías telefónicas desarrollaron redes de este tipo, y el CCITT emitió un estándar, el X.25, que es el que se ha adoptado hasta hace muy poco.

1.2.3. Las redes de área local

Cuando empezó a ser habitual disponer de más de un ordenador en la misma instalación, apareció la necesidad de interconectarlos para poder compartir los diferentes recursos: dispositivos caros, tales como impresoras de calidad, un disco duro que almacenara los datos de la empresa, un equipo de cinta para realizar copias de seguridad, etc.

Nota

Con frecuencia se utiliza la sigla inglesa LAN (*local area network*) para identificar las redes de área local, y la sigla WAN (*wide area network*) para identificar las redes de gran alcance.

El diseño de las redes de área local siguió caminos completamente diferentes de los que se siguieron para las redes de gran alcance. En las redes de área local se necesita, habitualmente, establecer comunicaciones "muchos a uno" y "uno a muchos", lo que es difícil de conseguir con las redes de conmutación, pensadas para interconectar dos estaciones. Para este tipo de redes es más adecuada la **difusión con medio compartido**, en que los paquetes que salen de una estación llegan a todo el resto simultáneamente. En la recepción, las estaciones los aceptan o ignoran dependiendo de si son destinatarias delos mismos o no.



Difusión con medio compartido

Se habla de *difusión* porque los paquetes se difunden por todos lados, y de *medio compartido* porque esta última se lleva a cabo sobre un medio común que las estaciones comparten.

1.3. Arquitecturas de protocolos

De la década de los sesenta datan también los primeros estándares de arquitecturas de protocolos. Conviene tener presente que el intercambio de información entre ordenadores tiene toda una serie de implicaciones, entre las que se encuentran las siguientes:

- Aspectos eléctricos: los cables, los conectores, las señales, etc.
- La manera de agrupar los bits para formar paquetes y la de controlar que no se produzcan errores de transmisión.

ANOTACIONES

- La identificación de los ordenadores dentro de la red y la manera de conseguir que la información que genera un ordenador llegue a quien se pretende.

Atacar todos estos aspectos de una manera global no es viable: demasiadas cosas y demasiado diferentes entre sí. Por ello, ya desde el principio, se desarrollaron modelos estructurados en niveles: en cada nivel se lleva a cabo una tarea y la cooperación de todos los niveles proporciona la conectividad deseada por los usuarios.

Conviene considerar que, en la época que nos ocupa, la informática estaba en manos de muy pocos fabricantes e imperaba la filosofía del servicio integral: cada fabricante lo proporcionaba todo (ordenadores, cables, periféricos, sistema operativo y software). Por tanto, cuando una empresa se quería informatizar, elegía una marca y quedaba vinculada a la misma para toda la vida.

Nota

Hablamos de empresas como IBM (International Business Machines) o DEC (Digital Equipment Corporation). Cuando estas empresaas se propusieron ofrecer conectividad entre sus equipos, local o remota, también lo hicieron aplicando la filosofía de la separación por niveles: IBM desarrolló la arquitectura SNA (*system network arquitecture*) y DEC, la DNA (DEC *network arquitecture*). Eran dos modelos completos, estructurados en niveles, pero incompatibles entre sí, según la filosofía de la informática propietaria.

En la década de los setenta el panorama cambió radicalmente, sobre todo a causa de tres acontecimientos:

- La propuesta del protocolo Ethernet para redes locales.
- La aparición del sistema operativo Unix, que no estaba vinculado a ninguna marca comercial, compatible con todas las plataformas de hardware existentes.
- La invención de los protocolos TCP/IP, embrión de la actual Internet.

Se había allanado el camino para la aparición de los sistemas abiertos: no era preciso vincularse a ninguna marca para tenerlo todo. El

Nota

TCP/IP son las siglas de *transmission control protocol/ Internet protocol* (protocolo de control de transmisión/ protocolo de Internet).

hardware podía ser de un proveedor, el sistema operativo de otro, las aplicaciones de otro y los protocolos, públicos.

TCP/IP nació a partir de un encargo de la DARPA a la comunidad científica americana para obtener una red mundial que fuera reconfigurable con facilidad y de forma automática en caso de destrucción de algún nodo o de algún enlace.

La pila TCP/IP era una jerarquía de protocolos que ofrecía conectividad y, a pesar de tener poco que ver con las que ya existían, constituía una opción más en el mercado. Ante una oferta tan grande y dispar de protocolos, la ISO (Organización Internacional de Estandarización, International Organization for Standardization) y el CCITT propusieron un nuevo modelo que intentaba reunir de algún modo todo lo que ya se había propuesto y que pretendía ser completo, racional y muy bien estructurado (la TCP/IP tiene fama de ser una pila de protocolos anárquica), con la intención, por tanto, de que se convirtiera en un modelo de referencia. Es la conocida como ***pila de protocolos OSI*** (*open systems interconnection*).



Internet, que nació y creció en las universidades, se empezó a popularizar en la década de los noventa, a medida que quienes conocían la Red la iban "enseñando", y su eclosión se produjo cuando saltó al mundo de la empresa, en todas sus vertientes: como escaparate de productos o como canalizador de contactos comerciales.

Sin embargo, el origen universitario de la Red ha marcado su evolución en muchos sentidos. Por ejemplo, el modelo cliente/servidor de aplicaciones distribuidas. Es un modelo sencillo y, al mismo tiempo, potente, y casi todas las aplicaciones que se utilizan en Internet lo siguen. El Telnet, o apertura de sesión remota, la transferencia de ficheros (FTP), el correo electrónico y, sobre todo, el WWW (World Wide Web) constituyen ejemplos claros de aplicaciones que siguen este modelo. Las dos primeras han caído un poco en desuso, pero tanto el correo como el WWW son las ac-

ANOTACIONES

tuales estrellas en Internet. Tímidamente, aparecen nuevas propuestas de aplicaciones; sin embargo, el WWW, que nació como un servicio de páginas estáticas enlazadas con hiperenlaces, se está convirtiendo en la interfaz de usuario de toda la Red, puesto que en la actualidad se utiliza para servir páginas dinámicas (se crean en el momento en que se sirven), e, incluso, código que se ejecuta en el ordenador cliente (*applets*).

1.4. La digitalización de la red telefónica

En este momento tenemos dos redes completamente independientes entre sí, pero de alguna manera superpuestas:

- Una red analógica, con conmutación de circuitos, pensada para voz.
- Una red digital, con conmutación de paquetes, pensada para datos.

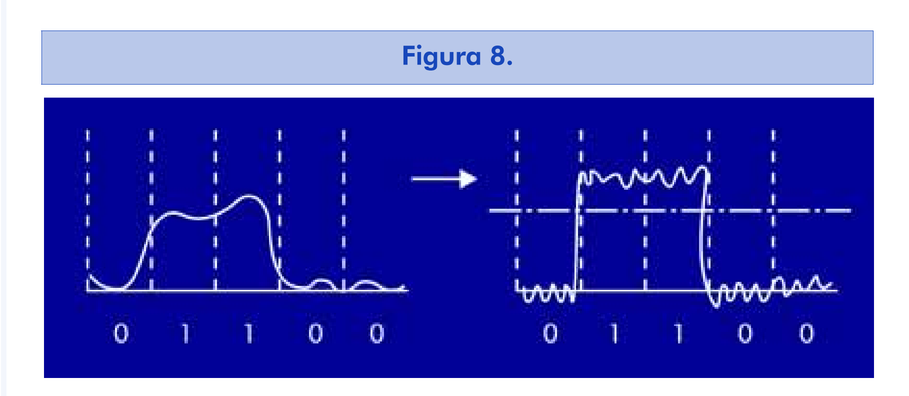
La red telefónica, tal como la hemos descrito hasta ahora, es completamente analógica: la señal electromagnética que viaja desde un teléfono hasta otro es analógica (varía continuamente y en cualquier momento puede adoptar cualquier valor) y los circuitos electrónicos que componen la red también lo son.

Los enlaces entre centrales de la red telefónica se llevaban a cabo con señales analógicas con muchos canales multiplexados en frecuencia y, en ocasiones, debían recorrer grandes distancias. La atenuación de la señal inherente a la distancia que era preciso recorrer debía corregirse por medio de repetidores que la amplificaban, lo que aumentaba el ruido presente en la línea. A menudo, la señal recibida era de una calidad muy baja porque la transmisión analógica no permite eliminar el ruido y las interferencias en la recepción. No hay manera de saber con exactitud qué se ha enviado desde el origen y qué es ruido añadido.

En 1972, se hicieron públicos los primeros resultados del tratamiento digital de la señal aplicado a audio, básicamente orientado a su almacenamiento. El CD estaba viendo la luz. Convertir un sonido (una magnitud física que puede adoptar cualquier valor en cualquier momento) en una serie de 0 y 1 (dos únicos valores, conocidos) permitía corregir con facilidad cualquier ruido añadido.

ANOTACIONES

Figura 8.



En el caso de la señal analógica, viendo la señal recibida, no se puede deducir cuál ha sido la señal emitida. En cambio, en el caso de la señal digital, como se conocen los valores enviados, se establece un umbral en el punto medio entre los dos valores y se decide que todo lo que esté por encima corresponde a un 1 y todo lo que esté por debajo, a un 0. Si el ruido que se ha añadido es superior a la diferencia entre el valor original y el umbral, se produce un error de recepción: se decide que se había enviado el valor equivocado. Las técnicas para luchar contra este tipo de errores se verán más adelante.

El descubrimiento del procesado digital de la señal, así como sus aplicaciones en los campos del sonido y la imagen, ha constituido un hito capital en el mundo de las comunicaciones. Básicamente, ha permitido reducir drásticamente el efecto del ruido, lo que ha posibilitado, por un lado, incrementar la calidad de recepción de las señales y, por el otro, aumentar la velocidad de transmisión con los mismos medios.

Las compañías telefónicas empezaron a sustituir los enlaces internos (entre centrales) por señales digitales, pero manteniendo el bucle de abonado (línea y terminal) analógico. La digitalización de la señal de sonido se lleva a cabo dentro de la central local, después del filtro de 4 kHz, y se vuelve a pasar a analógica en la central correspondiente al otro extremo de la comunicación. La digitalización ha hecho cambiar sustancialmente los procesos de conmutación: ahora debe trabajarse con bits y, por tanto, las centrales electromecánicas deben sustituirse por ordenadores.



La digitalización de la parte interna de la red de voz hizo que, de algún modo, las dos redes, la telefónica y la de datos, confluyeran: los enlaces digitales entre centrales se utilizaban indistintamente para paquetes de datos y para transmisiones de voz.

ANOTACIONES

1.4.1. La red digital de servicios integrados

Una vez digitalizada la red telefónica, el paso siguiente debía ser llevar la transmisión de bits hasta las casas. Ello permitía, por un lado, ofrecer a los usuarios en su casa la transmisión de datos además de la tradicional de voz y, por otro, ofrecer a los abonados un abanico de nuevos servicios asociados a una comunicación enteramente digital de extremo a extremo. Este servicio de transmisión digital por medio de la red telefónica se conoce como red digital de servicios integrados (RDSI). Ofrece dos canales independientes de 64 kbps, que permiten hablar y conectarse a Internet simultáneamente, o, con el hardware adecuado, aprovechar los dos canales juntos para navegar a 128 kbps.

Nota

La red digital de servicios integrados (RDSI) corresponde a las siglas en inglés ISDN (*integrated services digital network*).

1.5. La banda ancha

El uso de la red telefónica para transmitir datos tiene una limitación importante por lo que respecta al máximo de bits por segundo permitidos y las redes específicas de datos son muy caras para el uso doméstico. Desde la década de los noventa, se han estudiado maneras de llevar hasta las casas o las empresas un buen caudal de bits por segundo (banda ancha) a un precio razonable, de manera que las nuevas aplicaciones multimedia se puedan explotar al máximo.

Para conseguir esta banda ancha, se han seguido dos caminos completamente diferentes:

- Se han promovido cableados nuevos con fibra óptica que permitan este gran caudal, con frecuencia implementados por empresas con afán competidor contra los monopolios dominantes. Estas redes se aprovechan para proporcionar un servicio integral: televisión, teléfono y datos.

- Las compañías telefónicas de toda la vida han querido sacar partido del cableado que ya tienen hecho y, por ello, se han desarrollado las tecnologías ADSL, que permiten la convivencia en el bucle de abonado de la señal telefónica y una señal de datos que puede llegar a los 8 Mbps.

ANOTACIONES

Nota

La frontera entre banda estrecha y banda ancha no está muy clara. Los 128 kbps de la RDSI se consideran banda estrecha y, hay quien califica de banda ancha a los 256 kbps de la ADSL básica.

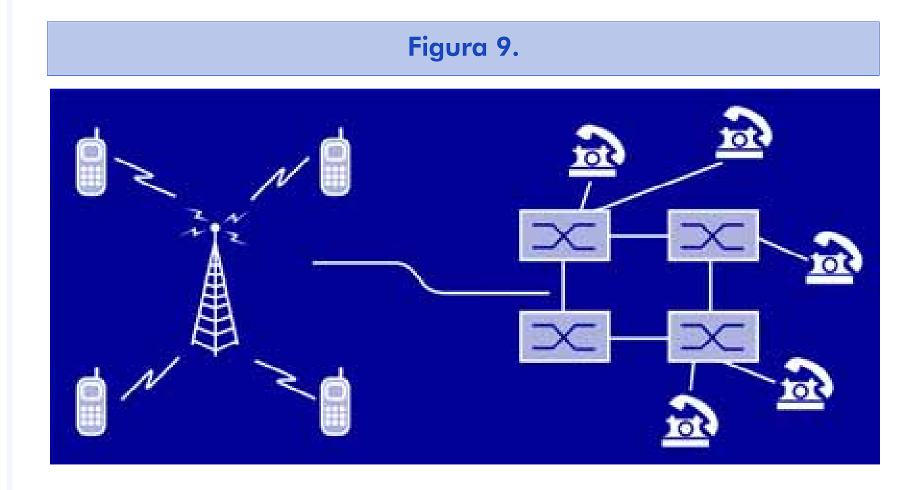
Realmente, se considera banda ancha a partir de 1 Mbps.

1.6. La telefonía móvil

La telefonía móvil, todo un fenómeno sociológico de finales del siglo xx, ha vivido una evolución fulgurante: en menos de veinte años, ha pasado de la nada a constituir una tecnología de uso diario para más de un 70% de la población.

Desde el punto de vista de sistema de comunicación, debemos ver los móviles como una extensión de la red telefónica convencional.

Figura 9.



El sistema GSM, que constituye el actual estándar europeo, permite el acceso a la red de voz, cambiando el bucle de abonado: en lugar de ser un cable, es un enlace radioeléctrico entre una antena y el móvil. Se trata, por tanto, de una red de conmutación de circuitos y se continúa fijando la tarifa por tiempo de conexión.

El estándar GPRS permite el transporte de bits, pagando por tráfico en lugar de por tiempo. Por tanto, es aproximadamente el clónico de las redes de datos con hilos.

ANOTACIONES

El estándar UMTS, en la actualidad todavía en la fase previa a su lanzamiento comercial, permite transferencias del orden de megabits por segundo, necesarias para disponer de aplicaciones multimedia en el móvil. Sin embargo, requiere nuevas antenas y terminales.

ANOTACIONES

2. Arquitecturas de protocolos: el modelo OSI

Como ya hemos comentado, cuando el CCITT y la ISO propusieron la torre OSI, en el mercado había muchas arquitecturas de protocolos, unas propietarias, otras abiertas, pero todas diferentes. La torre OSI pretendía ser un modelo básico de referencia, un marco para el desarrollo de estándares que permitieran la interoperabilidad completa. Diferentes razones han hecho que este modelo, así como las normas que del mismo se derivan, no hayan tenido la repercusión que se esperaba, entre las que destacan las siguientes:

- La complejidad del modelo, innecesaria en muchos casos.
- La complejidad de las normas desarrolladas a partir del modelo.
- El impulso del modelo Internet y la simplicidad de sus estándares.

A pesar de que el modelo OSI no se haya impuesto en los desarrollos, es muy útil como referencia para explicar qué debe hacerse y cómo. El hecho de que sea tan completo y cartesiano lo hace muy interesante para la pedagogía de los conceptos básicos de redes, y las arquitecturas que en realidad se utilizan se explican estableciendo una relación constante con el modelo OSI. Por ello, en este apartado explicamos los siete niveles de la torre OSI. A partir del módulo siguiente, sin embargo, nos centraremos en la arquitectura TCP/IP, la que constituye la Red Internet.

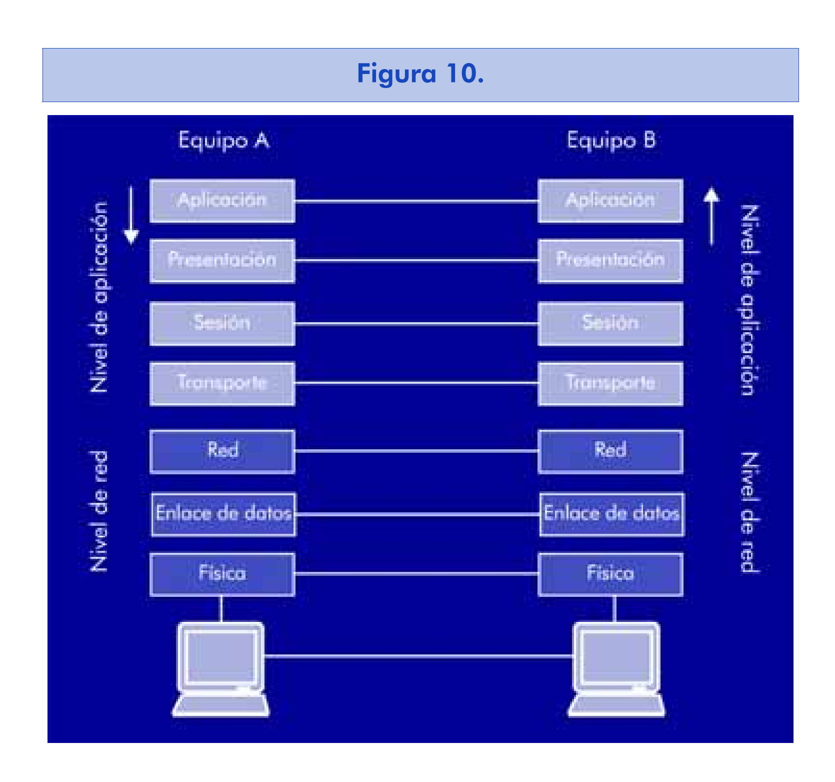
2.1. Definición



El modelo básico de referencia OSI, o simplemente modelo OSI, afronta el problema de las comunicaciones de datos y las redes informáticas dividiéndolo en niveles. Cada participante de la comunicación incorpora como mínimo uno de los mismos, y los equipos terminales los incorporan todos.

ANOTACIONES

Figura 10.



Los niveles de la torre se comunican en dos direcciones:

- **Horizontal**. La comunicación horizontal sólo se da entre niveles homónimos. Se podría pensar –y de hecho es así– que todo el nivel constituye un único sistema distribuido que tiene un representante en cada uno de los equipos. Un **protocolo de nivel** ***i*** (en el que *i* es el identificador del nivel correspondiente) especifica el formato, el significado y la temporización de la información que circula entre los miembros de este sistema distribuido.

- **Vertical**. La comunicación vertical sólo se da entre niveles adyacentes de un mismo sistema. Este tipo de comunicación posee un carácter totalmente local; es decir, puede materializarse por mecanismos de software (llamadas a liberías, comunicación entre procesos, etc.). De manera genérica, denominaremos estos mecanismos *servicio de nivel i* (en el que *i* es el identificador del nivel que proporciona el servicio, e *i* + 1, el nivel que lo utiliza).

2.2. Los protocolos

Con los protocolos se pretende la intercomunicación de entidades situadas en diferentes máquinas. Entendemos por ***entidad*** un sistema electrónico y/o informático, ubicado dentro de un nivel del modelo OSI,

ANOTACIONES

que, en combinación con las otras entidades del mismo nivel situadas en otros sistemas, forma un todo (un sistema distribuido).

Por tanto, la **especificación del protocolo** que utilizamos debe llevarse a cabo en un estándar claramente definido que permita a desarrolladores que no trabajan juntos implementarlo de manera totalmente idéntica.

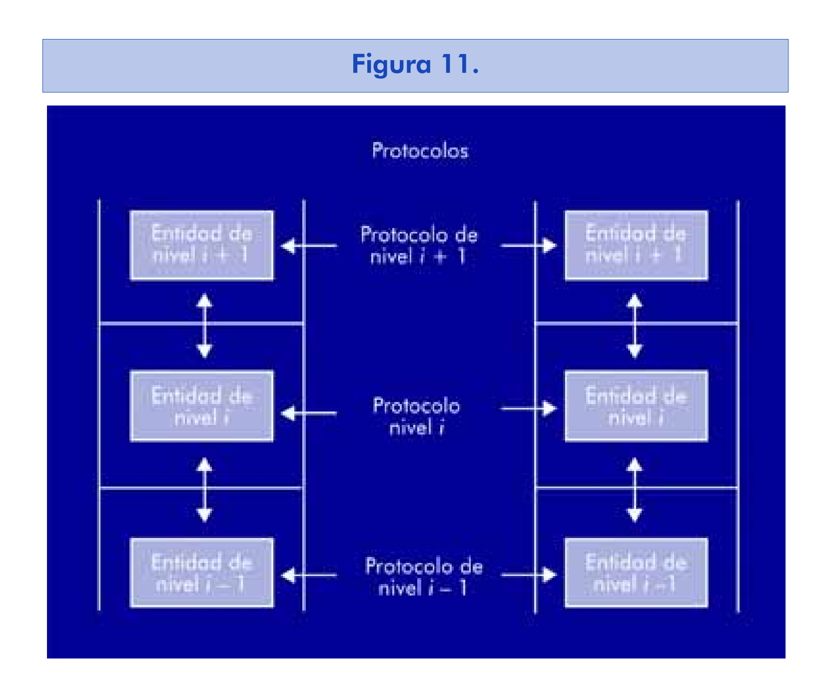
La recepción de una secuencia de bits en un momento inesperado o de una longitud incorrecta, o en una disposición imprevista, puede hacer que la entidad destinataria no reaccione correctamente y deje de inmediato el nivel (las dos entidades que lo forman) en una situación inestable.

Evidentemente, esta situción no se puede permitir. Por ello, la implementación del protocolo debe ser extremamente esmerada y, por consiguiente, también la especificación del estándar.



En un sistema encontramos tantos protocolos como niveles lo formen. Los sistemas a los que se conecte directamente deberán tener la misma especificación que los estándares para todos los niveles que implemente el protocolo.

Figura 11.



ANOTACIONES

2.3. Los servicios

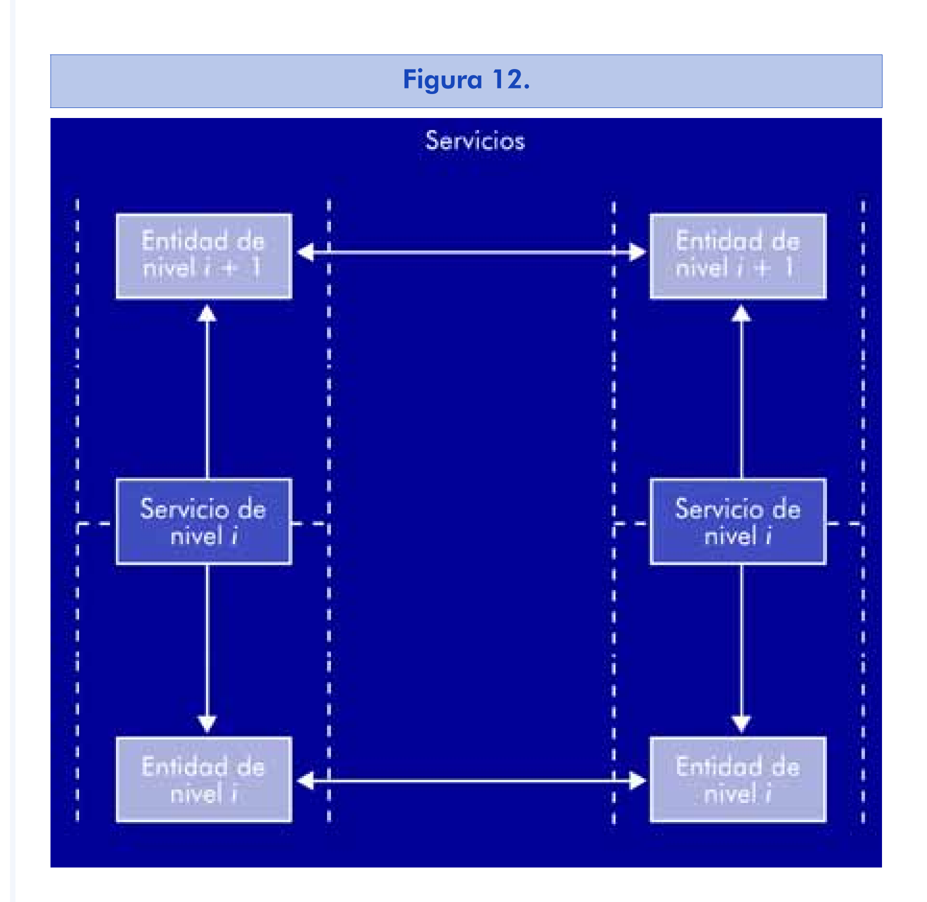
Nota

En terminología OSI se suele decir que los servicios no se especifican, sino que se describen.

La especificación de un servicio es siempre menos estricta que la de un protocolo. Por *servicio* entendemos la comunicación que se produce dentro de una misma máquina y, por consiguiente, dentro de un único ámbito de responsabilidad. La funcionalidad de las interfaces de cada uno de los niveles (y, por tanto, de las entidades que la implementan), la determinarán los estándares que utilicen; sin embargo, su especificación precisa no es relevante para los estándares involucrados. Cada sistema individual puede materializarlas de una manera u otra según convenga.

Sea como sea, la cantidad de papel que ocupa la descripción de un servicio siempre será muy inferior a la que ocupa la especificación de un protocolo.

Figura 12.



Actividad

Comentad las diferencias existentes entre protocolo y servicio.

2.4. Los siete niveles del modelo OSI

2.4.1. Nivel físico

El nivel físico se encarga de las tareas de transmisión física de las señales eléctricas (o electromagnéticas) entre los diferentes sistemas. Las limitaciones del nivel físico (equipos de transmisión y recepción, medios de transmisión, amplificadores, etc.) imponen otras al resto del sistema: por un lado, limitan la **velocidad de transmisión** (en bits por segundo) y, por otro, hacen aparecer una **probabilidad de error**, el porcentaje de bits erróneos que llegan a destino.

La primera limitación es casi insalvable partiendo de un **medio de transmisión** dado, puesto que los parámetros físicos de este último imponen un límite superior no superable por medio de una mejora tecnológica. Los medios de transmisión poseen una **capacidad** de transmisión acotada y la electrónica que utilizamos para llevar a cabo las transmisiones puede mejorar la velocidad de transmisión, pero no superar este límite. Esta limitación viene dada por el ancho de banda, o anchura del espectro eléctrico, que puede atravesar el medio de transmisión (doblar el ancho de banda significa que se puede doblar la velocidad de transmisión) y por la imposibilidad práctica de recibir la señal libre de cualquier interferencia.

Nota

En el nivel físico somos incapaces de corregir errores. Asumimos una probabilidad de error y encargamos al nivel superior su corrección.

2.4.2. Nivel de enlace

El nivel de enlace es el primero de la torre OSI que se basa en software, algoritmos y protocolos. Su misión principal es dar fiabilidad a la transmisión de las señales eléctricas o electromagnéticas que proporciona el nivel físico, lo que se puede conseguir si las cotas de error son inferiores al 1%. Se añaden bits adicionales a los que forman el mensaje para poder detectar errores de transmisión y pedir su retransmisión. Para ello, es preciso conferir una estructura a los bits: se agrupan en pequeños bloques denominados ***tramas***, que contienen los bits de mensaje, los bits añadidos para detectar errores y diferentes campos de control, tales como el número de trama.

Nota

El hecho de que las tramas sean pequeños bloques de bits minimiza la probabilidad de que haya muchos bits erróneos dentro de los bloques.

El transmisor calcula estos bits adicionales a partir del resto por medio de una operación que el receptor conoce y aplica igualmente. Si

ANOTACIONES

el receptor detecta una discrepancia entre los bits adicionales (redundantes) y los que ha calculado a partir del resto, detecta que el bloque es erróneo y pedirá su retransmisión.



La adición de los bits redundantes y su comparación en recepción se denomina *detección de errores*. Los procedimientos de corrección a partir de dicha detección se conocen como *control de errores*.

Además del control de errores, el nivel de enlace lleva a cabo otra tarea importante: el *control de flujo*.

El receptor debe procesar las tramas a medida que las recibe. En algunos casos, este proceso comporta un gasto de tiempo mínimo, teniendo en cuenta la velocidad de transmisión (por ejemplo, guardar los datos en disco); sin embargo, puede haber casos en que este proceso sea costoso. En esta situación, el receptor necesita un mecanismo que notifique al transmisor que debe detener momentáneamente la transmisión con el objetivo de disponer del tiempo necesario para llevar a cabo esta tarea.

El nivel de enlace no sólo sirve para controlar líneas punto a punto, sino también para controlar líneas compartidas por diferentes terminales (redes de área local).

2.4.3. Nivel de red

El nivel de red es el que permite que pueda haber más de dos máquinas involucradas en las interconexiones. Si sólo se tuviese el nivel de enlace, esto no sería posible. El nivel de enlace se ocupa de que los bits lleguen de una lado a otro, por lo tanto, sólo permite interconectar dos máquinas. Para poder interconectar más de dos máquinas, necesitamos identificarlas y conectarlas de alguna manera. Ésta es la tarea del nivel de red.

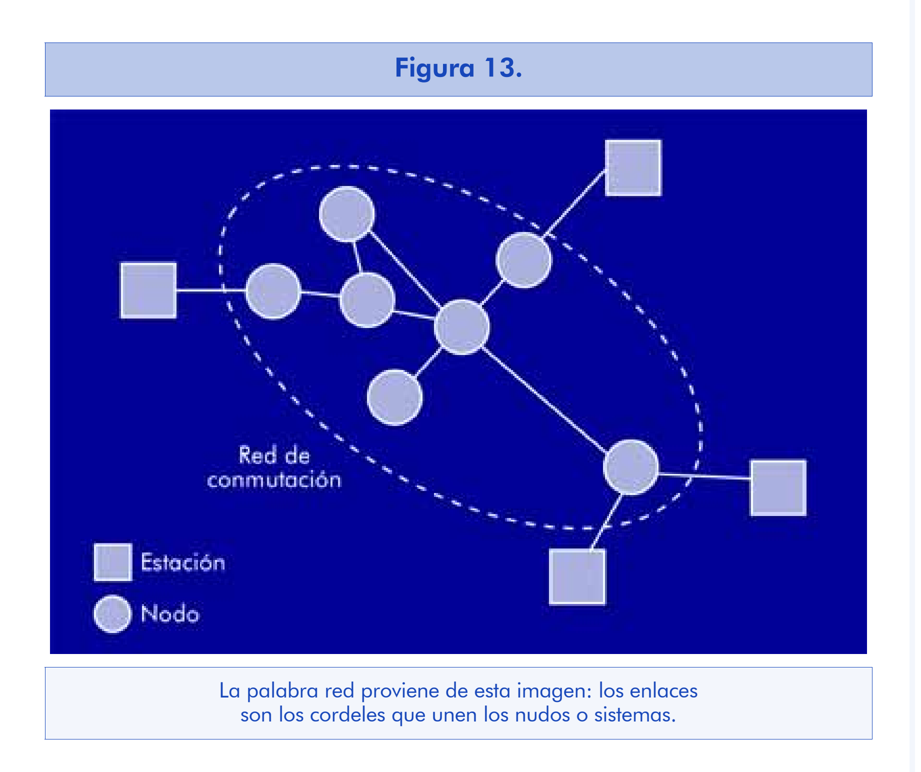
Ya hemos visto que las redes de conmutación de paquetes constituyen el tipo de red más eficiente para transmitir datos desde diferentes

ANOTACIONES

puntos de vista: uso de recursos, coste, capacidad de mantener distintas conexiones simultáneas, etc. El modelo OSI, por tanto, sólo habla de redes de conmutación de paquetes.

En el nivel de red se distingue entre estaciones terminales y nodos de conmutación:

Figura 13.



La palabra red proviene de esta imagen: los enlaces son los cordeles que unen los nudos o sistemas.

Los nodos de conmutación disponen de diferentes enlaces hacia otros nodos o hacia terminales, y son los que permiten que los paquetes viajen por la red desde una estación terminal a otra.

Existen dos tipos de redes de conmutación de paquetes:

- Redes que funcionan en modo **datagrama**. Podríamos decir que este tipo de redes son las básicas, puesto que incorporan la funcionalidad mínima para que un grupo de nodos y de terminales interconectados puedan hacer pasar información de un punto a otro.

  El problema de las redes en modo datagrama radica en la dificultad de garantizar la entrega correcta y completa de la información, puesto que los diferentes paquetes que forman la transmisión no mantienen un vínculo conocido por la red. Los paquetes pueden llegar fuera de orden, duplicados, o incluso se pueden perder sin

ANOTACIONES

que la red pueda hacer gran cosa al respecto. Se deja al terminal receptor la responsabilidad de restaurar los posibles daños que haya tenido el paquete durante la transmisión.

- Redes que funcionan en modo **circuito virtual**. Estas redes pueden garantizar que la entrega de los paquetes sea correcta y completa, y lo hacen aportando el concepto de **conexión** propio de las redes de conmutación de circuitos. Es el circuito virtual. Este último permite agrupar los paquetes relacionados de manera que el receptor los recibe correctamente sin problemas de orden, duplicación o pérdida.

La **asignación de direcciones** es uno de los conceptos básicos del nivel de red. Le permite, como sistema distribuido pero único, decidir cuál de los múltiples terminales es el destinatario final de cada paquete.

El **direccionamiento** constituye el procedimiento que permite a este sistema distribuido conducir la información por los diferentes nodos de origen a destino, minimizando el trayecto y el tiempo de tránsito, optimizando recursos, etc.

2.4.4. Nivel de transporte

El nivel de transporte permite una conexión fiable sobre cualquier tipo de red (fiable o no). En las redes de conmutación de paquetes en modo datagrama es donde este nivel revela su importancia, puesto que es el responsable de controlar las posibles deficiencias de las transmisiones.

La función principal de este nivel consiste en asegurar la calidad de transmisión entre los terminales que utilizan la red, lo que implica recuperar errores, ordenar correctamente la información, ajustar la velocidad de transmisión de la información (control de flujo), etc.

2.4.5. Niveles de sesión, presentación y aplicación

Estos tres niveles se suelen explicar de manera conjunta, puesto que existen pocos ejemplos prácticos de protocolos de sesión y de presentación. Además, la arquitectura Internet delega todos los trabajos

por encima de transporte a la aplicación. No obstante, en el modelo OSI están definidos como tres niveles diferentes e independientes, con atribuciones propias.

El nivel de sesión es, en teoría, el encargado de gestionar las conexiones de larga duración, la recuperación de caídas de red de manera transparente y los protocolos de sincronía entre aplicaciones.

El nivel de presentación se encarga de conseguir que las diferentes plataformas (sistemas operativos, procesadores, etc.) se puedan entender al conectarse por medio de una misma red. Dicho de otra manera, soluciona el problema de la hetereogeneidad definiendo una manera universal de codificar la información. Dicha codificación puede tener propiedades de eficiencia (por medio de la compresión, por ejemplo), propiedades de confidencialidad (por medio de la criptografía), etc.

En el nivel de aplicación residen los programas. En este nivel podemos encontrar servidores, clientes que acceden a estos últimos, aplicaciones que trabajan según un modelo simétrico (*peer-to-peer*), etc.

Actividad

Asignad los diferentes niveles de las redes que conocéis a las funciones explicadas en este apartado.

ANOTACIONES

II. Redes de área local

3. Las redes de área local

Una red de área local es un sistema que permite la interconexión de ordenadores que están próximos físicamente. Entendemos por *próximo* todo lo que no sea cruzar una vía pública: una habitación, un edificio, un campus universitario, etc.

En el momento en que una red debe cruzar una calle, o una vía pública en general, es preciso que una compañía de telecomunicaciones establezca la comunicación, puesto que son las únicas autorizadas para pasar líneas por zonas públicas.



Una definición más precisa de red de área local, prescinde de la distancia entre las estaciones y especifica que su carácter distintivo reside en que los mecanismos de enlace entre estaciones deben estar completamente bajo el control de la persona o entidad que establece dicha red.

Como comentábamos en la primera unidad, el objetivo que se perseguía cuando se propusieron las primeras redes de área local era compartir recursos entre diferentes ordenadores próximos (un sistema de almacenamiento masivo, una impresora o un dispositivo de conexión hacia el exterior, por ejemplo). Para este tipo de comunicaciones se propuso una filosofía de diseño basada en la difusión de tramas con medio compartido, de manera que cuando una estación pone una trama en el medio, el resto de estaciones puedan recibirla. Los receptores reales de la trama se la quedan y el resto, la ignora.

Nota

Las primeras redes de área local sólo permitían que uno de los ordenadores de la red (el servidor) ofreciera recursos al resto, que sólo podían actuar como clientes de este servidor, sin capacidad de ofrecer nada. De un tiempo a esta parte, el software de red que elaboran empresas como Novell, Microsoft o Apple permite que todas las estaciones puedan actuar como servidores y clientes al mismo tiempo.

ANOTACIONES

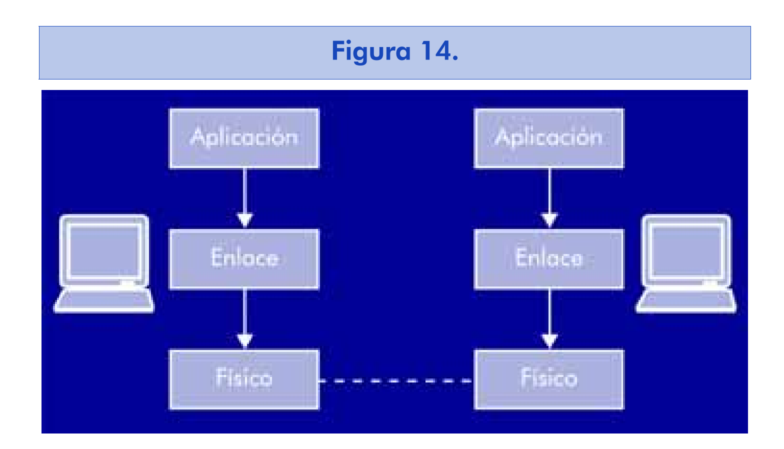
Últimamente y como veremos más adelante, se han aplicado técnicas de conmutación a las redes de área local, para conseguir mejorar su rendimiento.

Otra mejora importante ha sido la aparición de las redes de área local inalámbricas (*wireless* LAN), en las que el enlace entre estaciones no se lleva a cabo por medio de cables, sino por medio de enlaces radioeléctricos. Las ventajas de este tipo de enlaces, en cuanto a movilidad y facilidad de instalación, son evidentes.

En una red es imprescindible identificar los ordenadores que forman parte de la misma. Cuando un ordenador genera una trama para otro, además de los datos que le quiere enviar, le pone el identificador del ordenador (u ordenadores) destino y el suyo, para que quien reciba la trama pueda saber quién se la ha enviado.

Para construir una red local, se precisan básicamente dos cosas: hardware (tarjetas, cables, conectores) y un *sofware* que sea consciente de que existen diferentes máquinas conectadas y ofrezca los servicios necesarios para que las aplicaciones puedan aprovecharlo. Lo más lógico es que este software se integre en el sistema operativo y ofrezca a las aplicaciones la visión de la red como un recurso propio más. Estos recursos de hardware y software necesarios pueden ser analizados desde el punto de vista de la torre OSI, como se explicaba en la unidad anterior:

Figura 14.



Como se puede ver en la figura anterior, los niveles necesarios para implementar una red de área local son los dos inferiores (físico y enlace) y el superior (aplicación). A nivel de usuario no somos conscientes de esta subdivisión porque, como hemos dicho, el código que implementa los servicios asociados a los niveles está integrado en el sistema operativo de las estaciones.

Nota

Los niveles red, transporte, sesión y presentación tienen sentido en redes de área extensa, como veremos en las unidades siguientes.

ANOTACIONES

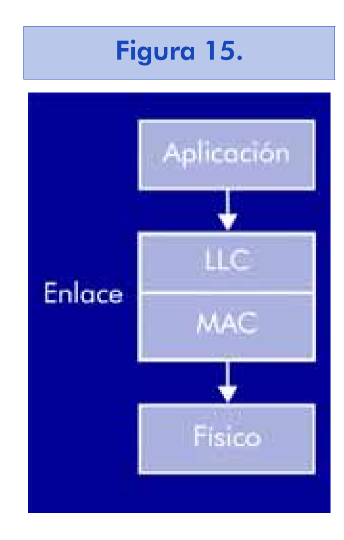
El nivel físico corresponde al hardware: a la tarjeta de red, a las señales electromagnéticas que viajan por el medio de transmisión, a los dispositivos que generan estas señales a partir de bits, etc.

El nivel de enlace, como ya sabemos, proporciona fiabilidad en el intercambio de tramas entre las estaciones: básicamente control de errores y control de flujo. Pero, por el hecho de usar un medio compartido, será necesario establecer mecanismos para que todas las estaciones puedan usarlo cuando lo precisen, pero sin molestarse. Si dos estaciones ponen tramas en el medio de transmisión de forma simultánea, éstas se mezclarán de manera que se convertirán en algo ininteligible. Esta situación se conoce como *colisión de tramas* y necesitamos mecanismos para controlar el acceso al medio compartido de manera que no se produzcan, o que si se producen, la red pueda recuperarse y seguir funcionando.

La inclusión de estos mecanismos en la torre OSI se podía llevar a cabo añadiendo un nivel más a la torre o, cómo al final sucedió, incluyéndolos en el nivel de enlace. Así, en contextos de área local, el nivel de enlace incluye dos subniveles:

- MAC (*medium access control* o control de acceso al medio), que se encarga própiamente de la política de acceso al medio
- LLC (*logical link control* o control del enlace lógico), que se encarga de los servicios típicos de enlace: control de errores y control de flujo.

Figura 15.



ANOTACIONES

4. Topologías de las LAN

Lo primero que caracteriza una red local es la manera en que se conectan las estaciones; es decir, la forma que adopta el medio compartido entre las mismas. Básicamente existen tres topologías posibles:

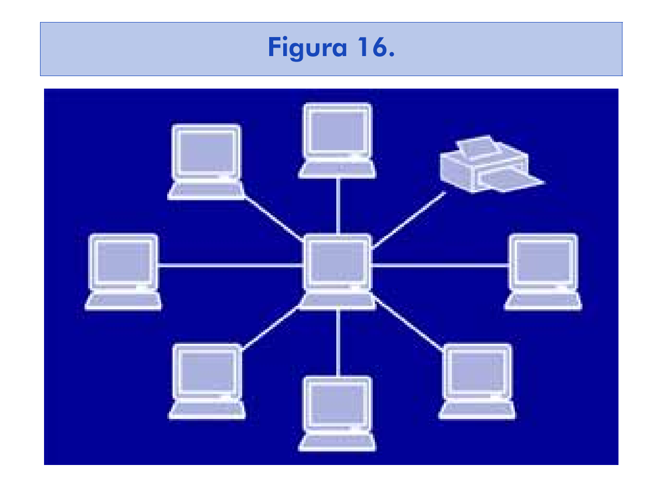
- Topología en estrella.
- Topología en bus.
- Topología en anillo.

4.1. Topología en estrella

La topología en estrella consiste en conectar cada ordenador a un punto central, que puede ser tan sencillo como una simple unión física de los cables.

Cuando un ordenador pone una trama en la red, ésta aparece de inmediato en las entradas del resto de ordenadores.

Figura 16.



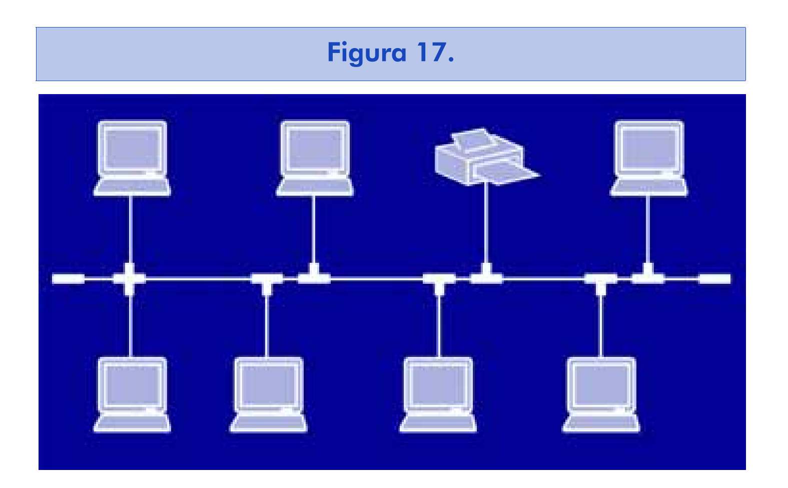
Aunque se han definido estándares para este tipo de redes, en la actualidad ya casi no existen, puesto que no aportan ninguna ventaja sobre el resto y sí muchos inconvenientes.

ANOTACIONES

4.2. Topología en bus

La topología en bus consiste en un cable al que se unen todas las estaciones de la red.

Figura 17.



Todos los ordenadores están pendientes de si hay actividad en el cable. En el momento en que un ordenador pone una trama, todos los ordenadores la cogen y miran si son el destinatario de la misma. Si es así, se la quedan, en caso contrario, la descartan.

Las primeras redes en bus utilizaban un cable coaxial grueso, conectores tipo BNC, y los ordenadors se conectaban al mismo con un dispositivo denominado *transceptor* (*transceiver*), que era exterior. Con posterioridad, apareció una nueva versión, con un cable más fino (*thin-ethernet*) y con unos transceptores más pequeños, de manera que se podían integrar en el adaptador de red y así no se veían.

Nota

Los caprichos de la electrónica exigen que el cable esté "tapado" en los dos extremos, para que los bits no se "pierdan". Ello se lleva a cabo con una resistencia de carga.

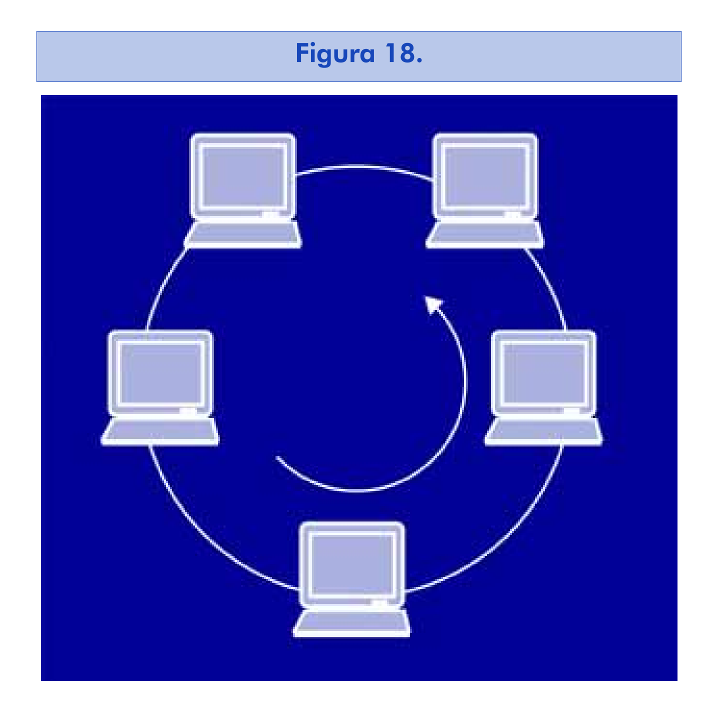
Cuando un ordenador pone una trama en el cable, ésta recorre el cable por completo en los dos sentidos hasta los extremos, donde es absorbida por los tapones.

ANOTACIONES

4.3. Topología en anillo

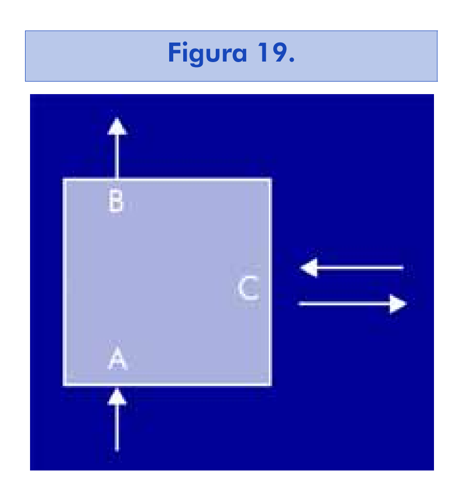
La topología en anillo consiste en conectar cada ordenador a dos más, de manera que se forme un anillo. Cuando un ordenador quiere enviar una trama a otro, ésta debe pasar por todos los ordenadors que haya entre ellos: la circulación por el anillo es unidireccional.

Figura 18.



El dispositivo que conecta el ordenador al anillo es el **repetidor**, un circuito con tres conexiones:

Figura 19.



- Conexión de entrada de tramas desde el anillo al ordenador (A).
- Conexión de salida de tramas desde el ordenador al anillo (B).

ANOTACIONES

- Conexión bidireccional, por la que pasan todas las tramas que entran y salen del ordenador (C).

El repetidor tiene tres modos de trabajo:

- Modo escucha: el repetidor toma las tramas que le llegan por A y las pone simultáneamente en B y C, para que continúen por el anillo y para que el ordenador reciba una copia de las mismas y la analice. Si es el destinatario de la trama, se la queda y, en caso contrario, la descarta.

- Modo transmisión: el ordenador envía información a la red. Pone una trama en C, de manera que cruza el repetidor y sale por B hacia el ordenador siguiente del anillo.

- Modo cortocircuito: las tramas que llegan por A se ponen directamente en B sin proporcionar una copia de las mismas al ordenador. Este modo sirve para que el anillo continúe funcionando si el ordenador correspondiente no está activo.

4.4. Pseudotopología de las redes inalámbricas

Hablar de topología en una red inalámbrica parece fuera de lugar, porque no 'vemos' ningún medio de transmisión. Pero en realidad el "éter" por donde viajan las ondas se considera un medio de transmisión, y si lo comparamos con las tres topologías descritas, vemos que se puede comparar a la topología en bus.

Nota

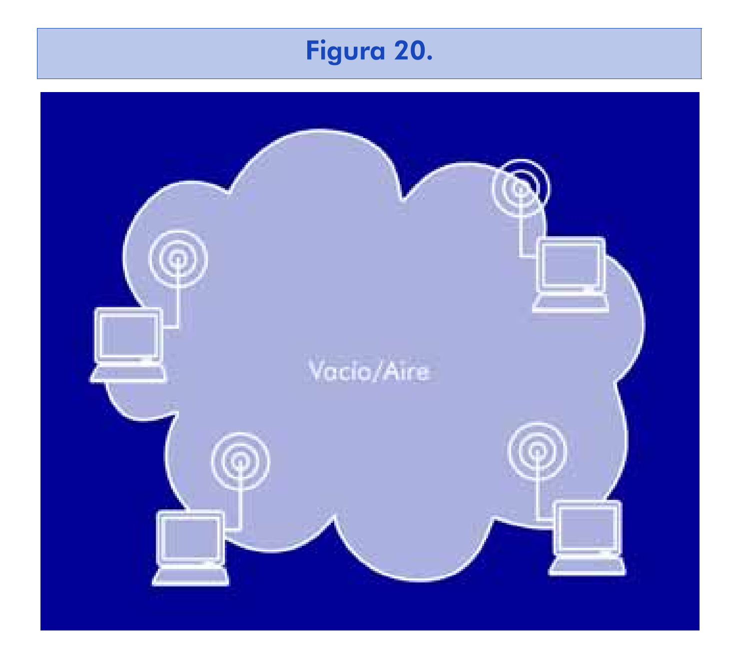
De hecho, las ondas electromagnéticas no necesitan ningún soporte físico para ser transmitidas. Se propagan en el vacío. Pero hasta que esto no fue demostrado, los científicos utilizaban el término "éter" para designar algo que se imaginaban que tenía que existir pero eran incapaces de ver.

En un anillo o en una estrella en realidad existen 'n' medios independientes que conectan una estación a otra (o al punto central), mien-

ANOTACIONES

tras que en un bus tenemos un sólo medio (un cable) al que se conectan todas las estaciones, de la misma manera que en una red inalámbrica tenemos un solo medio (el aire) donde las estaciones ponen sus tramas.

Figura 20.



La topología, como veremos más adelante, condiciona los mecanismos de acceso al medio que se pueden usar en una red local. En el caso de las redes inalámbricas esto es particularmente determinante.

ANOTACIONES

5. Cableado estructurado

Las topologías en bus y en anillo comportan un serio problema de cableado a la hora de implementarlas. Aunque es relativamente sencillo montar una red en bus o en anillo, es muy complicado mantenerla y ampliarla: cuando falla un cable o una conexión, la red entera deja de funcionar, y no es sencillo localizar el punto exacto donde se encuentra el fallo. Es preciso comprobar la red entera, lo que en numerosas ocasiones es complicado, puesto que los cables pueden pasar por falsos techos o conducciones de difícil acceso.

Este problema ha hecho pensar en un nuevo diseño de red más controlable: el cableado estructurado.

El cableado estructurado consiste en hacer una preinstalación de red similar a la de las redes telefónicas. A cada punto de trabajo se hacen llegar dos líneas: una para el teléfono y otra para los datos. Todos los cables llegan a una habitación, donde se establecen las conexiones: los cables de teléfono se direccionan hacia la centralita y los de los datos, hacia un dispositivo que permite la interconexión en red local.

En 1991 se publicó el EIA/TIA 568 sobre cableado de telecomunicaciones para edificios comerciales. El propósito de dicho estándar es:

- Ser universal, tanto en servicios soportados como en fabricantes compatibles.
- Ser base para el desarrollo de ottros estándares de comunicaciones (voz, imagen, LAN, WAN).
- Definir parámetros que permitan definir y establecer el cableado del edificio incluso antes que nadie lo ocupe. Se concibe el cableado como un servicio más del edificio (luz, agua, gas… y datos).

Nota

EIA/TIA:
Electronic Industries Association /Telecommunication Industry Association.

ANOTACIONES

El estándar especifica las señales a usar, así como los aspectos mecánicos de los cables, los conectores, los armarios, etc.

Por norma general, se realiza un cableado a dos niveles:

- Cableado **horizontal**: en cada planta (si es preciso cablear varias) se ponen cables desde un armario hasta los puntos terminales.

- Cableado **vertical**: desde cada armario de planta se ponen cables hasta una habitación del edificio donde se encuentran los dispositivos de red, los direccionadores (*routers*) hacia el exterior, la centralita telefónica, etc.

Nota

Hablando con propiedad diremos que una red tiene topología física en estrella y topología lógica en bus.

En cada planta necesitamos crear una red local en el punto donde confluyen los cables que provienen de cada una de las estaciones. Parece que una topología en estrella sería la más adecuada, pero como hemos comentado, tal como se había concebido, era la que ofrecía menos prestaciones. La solución es combinar las ventajas de la topología física en estrella con el funcionamiento de los buses o los anillos. O sea, usar para interconectar las estaciones un dispositivo, alojado en el armario de planta, que se comporte como un bus o como un anillo. En el caso del bus (la topología más utilizada actualmente), este dispositivo se conoce como concentrador o, en inglés, *hub*.

Lectura complementaria

Una buena aproximación al funcionamiento de los concentradores y los conmutadores la encontraréis en:

A.S. Tanenbaum (2003). *Redes de computadores*. Méjico: Pearson Educación.

Una topología así, donde el elemento central es un dispositivo activo que está simulando un dispositivo pasivo, llevó al desarrollo de las LAN conmutadas. El razonamiento es el siguiente: cuando el *hub* recibe una trama, para comportarse como un bus tiene que reenviarla hacia el resto de estaciones. Pero, el *hub* tiene capacidad de proceso: puede analizar la trama y, en particular, puede averiguar cual es su destinatario. Entonces, si el *hub* conoce los identificadores de las diferentes estaciones que tiene conectadas, puede enviar la trama únicamente a su destinatario, y así disminuir el número de tramas en la red, y, por tanto, aumentar la eficiencia. Los dispositivos que se comportan así se denominan conmutadores (en inglés, *switch*).

Por lo que se refiere al medio físico, se usan tanto pares de cobre trenzados como fibra óptica, aunque en mucha mayor medida los

ANOTACIONES

primeros por su menor coste para similares prestaciones. Se han especificado categorías de cables, cada cual con unas capacidades y unos requisitos mínimos a cumplir. Hoy en día el más usado es el cable categoría 5e, que pemite un ancho de banda de 100 MHz, el requerido para las LAN de alta velocidad, como Fast Ethernet y Gigabit Ethernet.



Los costes de instalación de un sistema de cableado estructurado son muy altos; pero su mantenimiento es muy simple y barato.

Si falla un cable, sólo falla una estación de trabajo, no toda la red, y, si falla toda la red, es que se ha estropeado el concentrador. Tanto un caso como el otro son muy rápidos de solucionar.

Actividad

Los que tengáis acceso a una instalación con cableado estructurado, estudiadla: observad las conexiones, los cables, los armarios de planta, los conmutadores, etc.

Nota

Hablaremos de FastEthernet y Gigabit Ethernet en el apartado siguiente.

ANOTACIONES

6. Control de acceso al medio

Dado que cualquier ordenador de la red puede poner tramas al medio compartido, es preciso establecer mecanismos de control que regulen este acceso de manera eficiente, justa y fiable.



El control de acceso al medio (MAC) es un mecanismo que decide qué estación tiene acceso al medio de transmisión para emitir una trama de información.

En general, los protocolos de acceso al medio se pueden clasificar en tres grandes grupos:

- Control de acceso al medio estático
- Control de acceso al medio dinámico (centralizado o distribuido)
- Control de acceso al medio aleatorio

Cada uno de estos tipos de accesos tiene ventajas e inconvenientes, y se apli-can a redes muy diferentes. De hecho, las políticas de acceso al medio están muy vinculadas a la topología utilizada. De este modo, en una topología en anillo, la manera más natural de controlar el acceso es por **paso de testigo** (*token passing*), que es un ejemplo de control dinámico distribuido. En la topología en bus, también se puede utilizar este sistema; sin embargo, está mucho más generalizado el uso de la técnica CSMA/CD, que es de tipo aleatorio. En las redes inalámbricas se usa una política de acceso al medio que es una combinación de control estático y aleatorio.

En esta unidad vamos a describir las dos políticas más comunes hoy en día en las redes cableadas: el paso de testigo y CSMA/CD.

Lectura complementaria

Podéis encontrar la definición de todos los tipos de control de acceso al medio en:

A.S. Tanenbaum (2003). *Redes de computadores.* Méjico: Pearson Educación.

Nota

CSMA es la sigla de Carrier Sense Multiple Access (acceso múltiple por detección de portadora) y CD es la sigla de Collision Detection (detección de colisiones).

6.1. Paso de testigo

Como decíamos, la política de paso de testigo es la más apropiada para las redes en anillo. Así pues, para describir su funcionamiento

ANOTACIONES

asumiremos que estamos en una red de esta topología. En inglés estas redes se denominan *token-passing ring*, literalmente “anillo con paso de testigo”.

El funcionamiento de la política de paso de testigo es el siguiente:

Se define una trama especial, el testigo. Cuando una estación lo recibe, tiene permiso para poner una trama propia en la red. Una vez esta trama ha dado toda la vuelta, y después de que sus destinatarios se hayan quedado una copia de la misma, la estación que la ha puesto la quita y libera el testigo que llegará a la estación siguiente del anillo. Esta estación repite el procedimiento: saca el testigo de la red y pone una trama suya o, si no tiene nada para enviar, pasa el testigo a la estación siguiente. Las estaciones que tengan información para transmitir deben esperar a tener el testigo para ponerla en la red.

Este mecanismo de control del medio permite con la misma facilidad la emisión de tramas tanto a una sola estación como a muchas. La trama recorre todo el anillo, por tanto todos los repetidores la ven pasar. Cada uno comprueba si en el campo “destinatario” de la cabecera de la trama aparece su identificador. En caso afirmativo, se queda una copia y la retransmite hacia la siguiente estación. En caso contrario la retransmite sin quedarse copia.

Las velocidades de trabajo de las redes en anillo con testigo están normalizadas: 4, 16 y 100 Mbps. Si se utiliza fibra óptica como medio de transmisión, la red, que se denomina FDDI (*fiber distributed data interface*), puede superar los 100 Mbps.

Nota

IEEE es la sigla del Institut of Electric and Electronic Engineers (Instituto de ingenieros eléctricos y electrónicos).

Las redes de paso de testigo fueron inventadas por IBM. Con posterioridad, el IEEE elaboró el estándar 802.5, que recogía toda la información existente sobre las mismas.

6.2. CSMA/CD

Como ya hemos comentado, CSMA/CD es una política de acceso al medio de tipo aleatorio, lo cual quiere decir básicamente que las estaciones no acceden al medio de una forma prefijada sino cuando quieren. De esta forma se consigue aumentar la eficiencia de la red

ANOTACIONES

con respecto a los sistemas de control estáticos. Obviamente hará falta controlar el caso en que dos estaciones quieran transmitir a la vez.

Nota

Los mecanismos de control del tipo estático se basan en repartir el acceso al medio entre las estaciones de forma fija. Si cuando a una estación le toca acceder al medio no tiene nada que transmitir, el lapso de tiempo asignado no puede ser aprovechado por otra y el medio queda desocupado.

De políticas de acceso al medio de tipo aleatorio hay varias, pero las "comercialmente útiles" son dos, CSMA/CD y CSMA/CA. La primera es la más indicada para redes con topología en bus (ya sea con un bus real, cableado, como con un *hub*, en un entorno de cableado estructurado). La segunda es la que se usa en las redes inalámbricas Wi-Fi, que como hemos comentado, tienen una topología asimilable a un bus.

Veamos en primer lugar como funciona CSMA, para luego describir la funcionalidad adicional de la detección de colisiones (CD).

La política de acceso CSMA (acceso múltiple por detección de portadora) funciona de la manera siguiente:

Los ordenadores escuchan constantemente el medio (miran si hay portadora). Cuando tienen una trama para transmitir, si detectan que no hay actividad en el medio, la ponen y, en caso contrario, esperan y siguen escuchando hasta que el medio queda libre, entonces transmiten su trama. Si no tienen nada para transmitir, cuando detectan una trama en el medio, la toman y la procesan.

Este algoritmo presenta un inconveniente claro: existe la posibilidad de que dos estaciones quieran enviar una trama en el mismo momento. Ambas escuchan el medio, no detectan actividad y emiten simultáneamente. Entonces se produce una colisión: las señales electromagnéticas se mezclan y el resultado es algo ininteligible. El control de errores que se efectúa en el subnivel LLC será el encargado de detectar dicha circunstancia y solicitar la retransmisión de las tramas que se han corrompido.

Podemos mejorar la política CSMA añadiéndole un procedimiento adicional: cuando una estación ya ha empezado a transmitir, si-

ANOTACIONES

gue escuchando el medio para detectar si se produce una colisión, en cuyo caso deja de transmitir inmediatamente, para reducir así el tiempo de colisión, y espera un tiempo aleatorio (es una manera de evitar una nueva colisión) para volver a empezar el proceso. Esta variante recibe el nombre de CSMA/CD, por la detección de las colisiones.

Este modo de trabajo marca la existencia de un parámetro muy importante en las redes CSMA/CD, que es la **longitud mínima de trama**.

Una trama en una red local CSMA/CD ha de ser suficientemente larga para que una colisión sea detectada antes de que finalice su transmisión. Esta longitud mínima de trama es función únicamente de la velocidad de transmisión de las señales en el medio. Este parámetro, a su vez, marca otro, también muy importante en el diseño de redes, como es el **diámetro de la red**, o distancia entre las estaciones más alejadas.

La red Ethernet es una topología en bus que utiliza la política de acceso al medio CSMA/CD y fija como longitud mínima de trama 512 bits. La inventó Xerox, junto con Intel y Digital, y el IEEE la elevó a la categoría de estándar: IEEE-802.3.

La red Ethernet trabaja a 10 Mbps. Con posterioridad, apareció la FastEthernet, que trabaja a 100 Mbps, y, más tarde, la Gigabit Ethernet que, como su nombre indica, trabaja a 1 Gbps.

Actividad

Que una red vaya a 10 Mbps no quiere decir que todas las conexiones se realicen a esta velocidad. Si tenéis acceso a una red de área local, por un lado, averiguad a qué velocidad funciona y por otro, haced una transferencia de información entre dos estaciones y medid la velocidad real a la que se ha realizado (el cociente entre el número de bytes transmitidos, multiplicado por ocho, y dividido por el tiempo que ha durado la transmisión, en segundos). Cuánto más grande sea el fichero que transmitáis, mejor. Podéis hacer la prueba bajo diferentes condiciones de trabajo, como por ejemplo con pocas estaciones activas, con muchas transmisiones simultáneas, etc. Comparad los valores obtenidos con la velocidad nominal de la red.

ANOTACIONES

III. TCP/IP

7. Estructura de protocolos en Internet

El modelo Internet gira en torno a los protocolos TCP/IP. IP es un protocolo que proporciona mecanismos de interconexión entre redes de área local y TCP proporciona mecanismos de control de flujo y errores entre los extremos de la comunicación.

No se trata de una arquitectura de niveles formal como la torre OSI, que ya hemos visto en la unidad 1. De hecho, podríamos considerar que el modelo de la red Internet consta sólo de cuatro partes o niveles; es decir, todo lo que hay por debajo del IP, el IP, el TCP y todo lo que hay por encima del TCP:

1) **Por debajo de IP.** A este nivel, en el entorno Internet, se le llama *nivel de red local* o, simplemente, *nivel de red*. Por norma general, está formado por una red LAN, o WAN (de conexión punto a punto) homogénea. Todos los equipos conectados a Internet implementan dicho nivel.

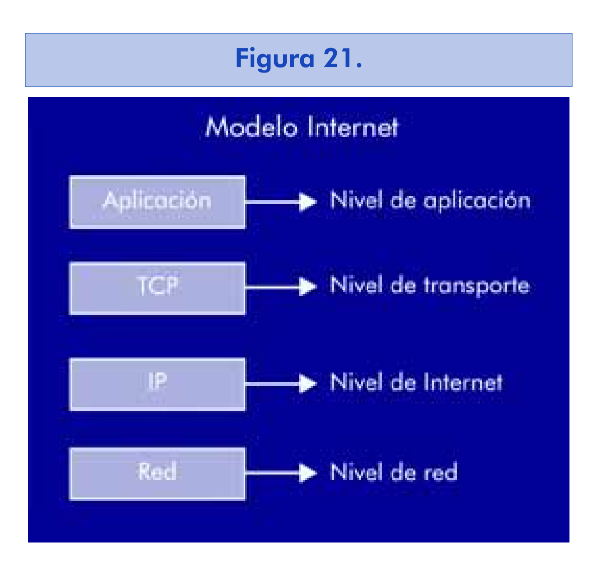
2) **Nivel IP o nivel Internet** *(nivel de Internetworking)*. Este nivel confiere unidad a todos los miembros de la red y, por consiguiente, es el que permite que todos se puedan interconectar, con independencia de si se conectan a la misma por medio de línea telefónica, ISDN o una LAN Ethernet. El direccionamiento y la asignación de direcciones constituyen sus principales funciones. Todos los equipos conectados a Internet implementan este nivel.

3) **Nivel TCP o nivel de transporte.** Este nivel confiere fiabilidad a la red. El control de flujo y de errores se lleva a cabo principalmente dentro de este nivel, que sólo es implementado por los equipos usuarios de la red Internet o por los terminales de Internet. Los equipos de conmutación (direccionadores o *routers*) no lo necesitan.

ANOTACIONES

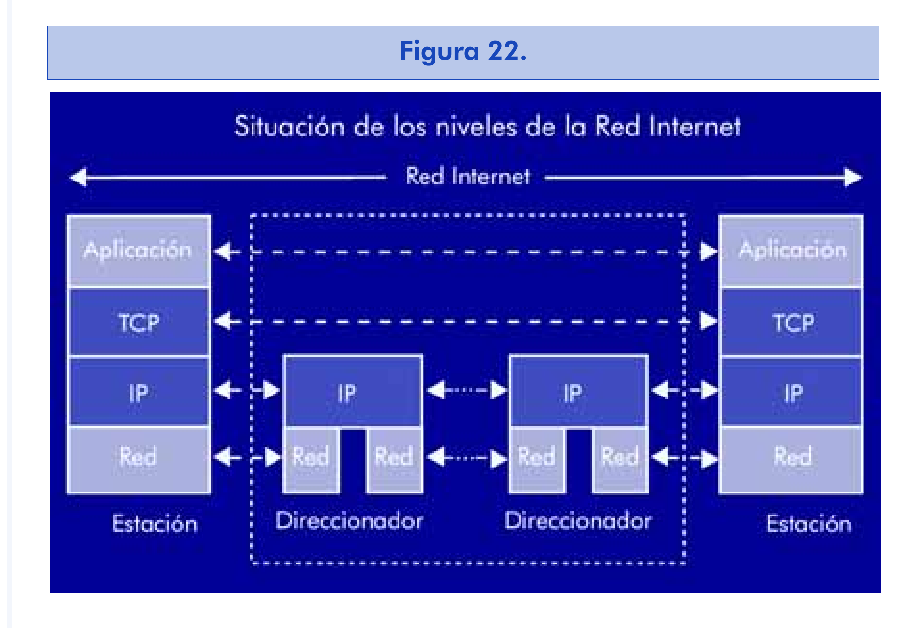
4) **Por encima de TCP. Nivel de aplicación**: Este nivel corresponde a las aplicaciones que utilizan Internet: clientes y servidores de WWW, correo electrónico, FTP, etc. Por ello se le denomina *nivel de aplicación*. Sólo es implementado por los equipos usuarios de la red Internet o los terminales de Internet. Los equipos de conmutación no lo utilizan.

Figura 21.



Es importante destacar que sólo los equipos terminales implementan todos los niveles; los equipos intermedios únicamente implementan el nivel de red y el nivel IP.

Figura 22.



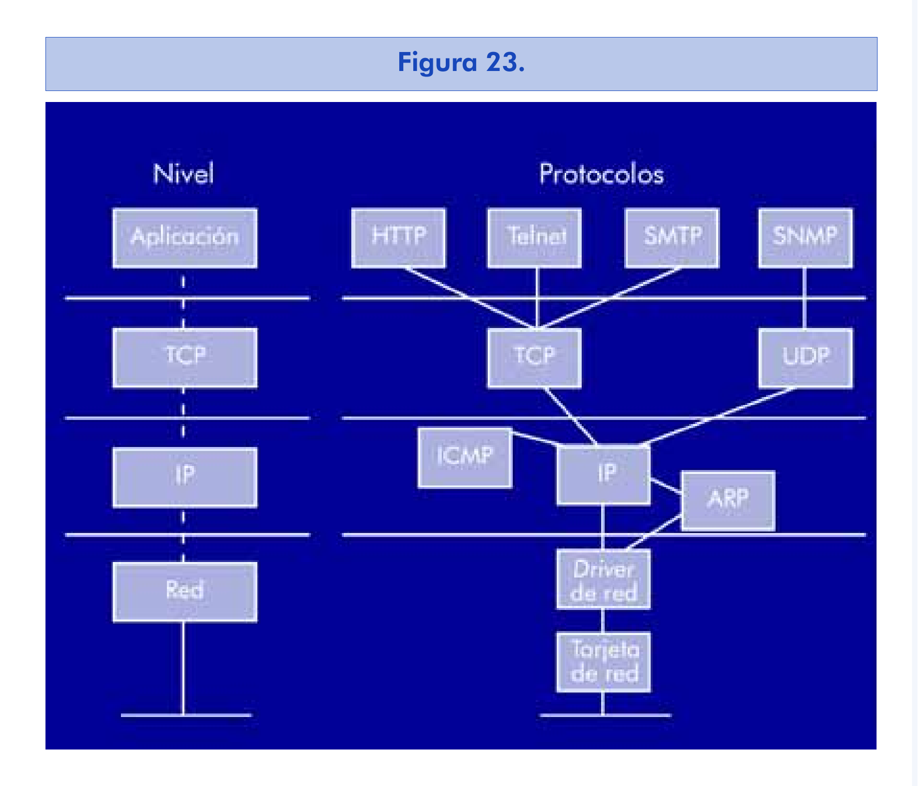
Internet se situa "encima" de las redes locales ya existentes. No redefine el concepto ni propone protocolos de red local nuevos.

ANOTACIONES

7.1. Protocolos de Internet

En los niveles intermedios existen otros protocolos complementarios, además de TCP e IP. La figura siguiente sería un mapa bastante completo de los protocolos que se usa en Internet:

Figura 23.



Como ya hemos comentado, el concepto *nivel* no existe en Internet. Este concepto se utiliza en otros modelos de red, como la OSI. No obstante, como es un concepto útil, lo utilizaremos para plantear el estudio de los diferentes protocolos de la manera siguiente:

a) En primer lugar, describiremos el IP y los protocolos que colaboran con él: ARP e ICMP.

b) Después, estudiaremos ciertos aspectos básicos del nivel de red para las diferentes posibilidades de conexión a Internet actuales: las LAN Ethernet y los accesos por línea punto a punto con el protocolo PPP o con ADSL.

Dejaremos para unidades posteriores el estudio del nivel de transporte y del nivel de aplicación.

ANOTACIONES

7.2. Encapsulamiento

En cualquier arquitectura de niveles (sea más o menos formal) en que exista una comunicación vertical dentro de cada máquina, los datos que se generan en el nivel superior (aplicación) atraviesan el resto de niveles para "salir" de la máquina por el nivel físico.

Cada uno de estos protocolos funciona con unas estructuras fundamentales que genéricamente se conocen como **PDU** (*protocol data units*). Sin embargo, en cada nivel se utilizan nombres diferentes para denominar lo que, de hecho, tiene funciones equivalentes. En el conjunto de protocolos Internet tenemos las siguientes PDU:

- Las PDU Ethernet o PPP se denominan ***tramas***.
- Las PDU del nivel de interconexión (IP o ARP) se suelen denominar *paquetes*, aunque las PDU ICMP se suelen denominar *mensajes*, seguramente porque viajan en paquetes IP.
- En el nivel de transporte, se habla de *segmentos* en TCP, y de *datagramas* en UDP.
- En niveles superiores que utilizan UDP, por norma general se utiliza la palabra *PDU* (SNMP-PDU, por ejemplo). En el caso del TCP, el servicio que proporciona a las aplicaciones es el flujo de bytes sin estructura (*byte stream*). Por tanto, el concepto *PDU* deja de tener sentido en el nivel superior a TCP.

El resultado de los diferentes encapsulamientos en cada nivel es que, cuando el nivel superior decide transmitir cierta información, se provoca una cascada de PDU que va descendiendo hasta el nivel inferior, que finalmente es el que transmite físicamente los bits que resultan del mismo.

Nota

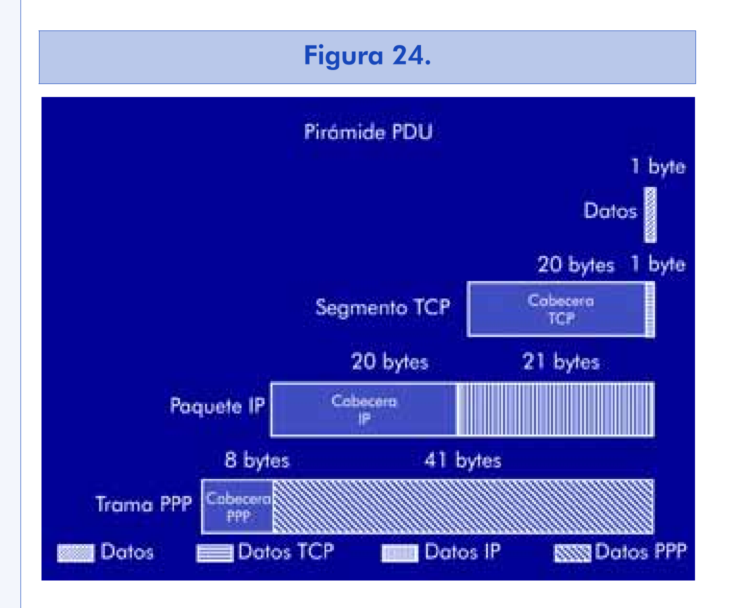
Para calcular la longitud de trama necesaria para transmitir un byte de información, veremos, a modo de ejemplo, la emulación de terminal que efectúa el programa `telnet` y el protocolo que lleva el mismo nombre.

ANOTACIONES

Imaginemos que pulsamos una tecla sobre la ventana del programa `telnet`. Esta acción llega en forma de un byte único al nivel TCP, que encapsula dicho byte en un segmento TCP. Este segmento tendrá una cabecera de 20 bytes y un contenido que será el byte correspondiente a la tecla pulsada. Estos 21 bytes se transportarán dentro de un paquete IP que, generalmente, formará un paquete con 20 bytes de cabecera más el contenido ya mencionado de 21 bytes. Este paquete de 41 bytes irá, a su vez, dentro de una trama que lo transportará por su soporte físico de transmisión. Si el soporte de transmisión es una línea telefónica con un módem y utilizamos PPP, el resultado puede ser que le añadamos 8 bytes más.

De todo ello resulta una trama de 49 bytes (8 + 20 + + 20 + 1) de longitud para transmitir uno solo. La figura siguiente muestra esta estructura:

Figura 24.



ANOTACIONES

8. El IP (*Internet protocol*)



IP y TCP son un par de protocolos bien compenetrados. El IP es un protocolo de interconexión de red orientado a datagrama. Por tanto, no dispone del concepto de circuito virtual, de manera que no es capaz de recuperar tramas perdidas, ni de garantizar que las tramas se entregarán en el orden correcto –puesto que los paquetes pueden seguir caminos diferentes y, por tanto, sufrir retardos diferentes–, ni que el ritmo de recepción sea el adecuado para que el receptor procese convenientemente los datos.

El IP es del tipo *best effort* (podríamos traducirlo como 'con la mejor intención', o 'quien hace lo que puede no está obligado a más'). Evidentemente, cuando utilizamos una red, no siempre podemos conformarnos con conseguir que la información llegue si la red puede. Aquí interviene el TCP, que es responsable de conseguir que la información llegue en las condiciones de fiabilidad deseadas.

Nota

Hay detractores acérrimos, así como defensores incondicionales de la filosofía *best effort*, aplicada a las redes y, seguramente, ambos grupos tienen una parte de razón. La red X.25 es un ejemplo casi opuesto a esta filosofía que proporciona a sus usuarios unos niveles de fiabilidad y flexibilidad razonablemente elevados, gracias a la utilización de la conmutación de paquetes con circuitos virtuales.

De hecho, la historia da la razón a los defensores de la filosofía *best effort*, puesto que el X.25 actualmente se considera una tecnología casi obsoleta, mientras que el IP está de moda. Las razones deben buscarse, como siempre, en el coste. La filosofía IP permite implementar redes con un coste mínimo: pensad que el TCP sólo precisa implementarse en los terminales de la red, y no enlos nodos de conmutación. Por tanto, los nodos de conmutación IP son mucho más simples que los de X.25.

Lectura complementaria

Una buena descripción de los algoritmos de direccionamiento usado en Internet se puede encontrar en:

W.R. Stevens (1994). TCP/IP Illustrated. Vol. 1 *The protocols.* Wilmintong: Addison-Wesley.

ANOTACIONES

8.1. Direcciones IP

Las direcciones IP son únicas para cada máquina. Para ser precisos, cada dirección es única para cada una de las interfaces de red IP de cada máquina. Si una máquina dispone de más de una interfaz de red, necesitará una dirección IP para cada una.

Las direcciones IP tienen una longitud de 32 bits (4 bytes).

Para representar una dirección, se suele escribir los 4 bytes en decimal y separados por puntos. Por ejemplo:

Nota

Él único objetivo de esta notación es la legibilidad humana. No se puede perder nunca de vista que una dirección IP son 32 bits.

212.45.10.89

La numeración en IP sigue una filosofía jerárquica. Cada dirección está formada por dos partes. Una corresponde a la red donde está la estación y la otra, a la propia estación.

Para conseguir que no haya ninguna dirección igual, Internet dispone de una organización denominada *Internet Network Information Center* o *InterNIC* que se dedica a esta tarea. En la actualidad, esta entidad delega la responsabilidad de la asignación de direcciones a entidades regionales. Las direcciones se asignan por grupos o redes, no individualmente.

Los tipos de redes que tienen cabida en Internet se distinguen por la cantidad de estaciones que pueden soportar, y son los siguientes:

1) **Las redes de clase A** reservan el primer byte como identificador de red y los tres restantes como identificadores de estación. El primer bit del primer byte vale 0, por tanto, en Internet sólo puede haber 128 redes de clase A (con 2^{24} estaciones cada una como máximo). Hace mucho tiempo que ya no queda ninguna para asignar.

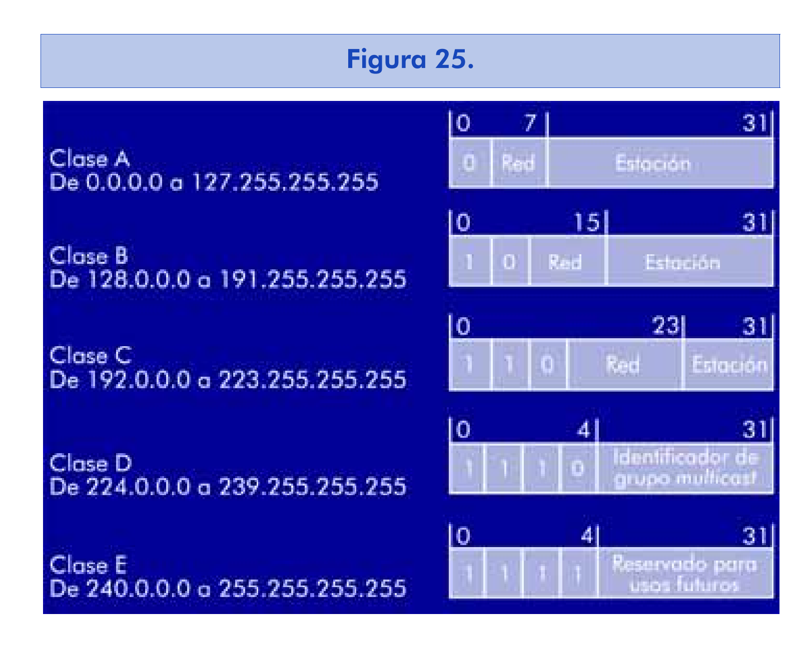
2) **Las redes de clase B** tienen 16 bits para cada campo; los dos primeros bytes del identificador de red valen 1 0, por tanto, hay 16.384 (2^{14}) redes de, como mucho, 65.536 estaciones. De clase B no queda ninguna para asignar.

ANOTACIONES

3) **Las redes de clase C** reservan 24 bits para el identificador de red (con los tres primeros bits 1 1 0) y los 8 restantes son para el identificador de estación.

Una vez que se conoce una dirección, es fácil saber si corresponde a una red de clase A, B o C, como se puede ver en la figura siguiente:

Figura 25.



La clase A está pensada para grandes empresas o corporaciones, con muchos terminales por identificar; la clase B, para corporaciones medianas; la clase C, para entornos mucho más pequeños; la clase D está destinada al tráfico multicast IP, y la clase E, de momento, no tienen ningún uso concreto. La dirección comentada con anterioridad (212.45.10.89) es de clase C.

Nota

Classless Inter-Domain Routing (CIDR)

El espacio de direcciones se está agotando a marchas forzadas y, si bien existen técnicas relacionadas con la seguridad que ayudan a mitigar este problema (como el NAT, *network address translation*) el hecho de utilizar el concepto de "clase" ha agravado el problema: algunas corporaciones que han reservado una red clase A o B y sólo han aprovechado una pequeña parte de la misma, han dejado sin un buen trozo del espacio de direcciones al resto de Internet.

ANOTACIONES

Hoy día, en lugar de proporcionar redes clase C (de las otras ya no quedan) lo que se hace es dar grupos mayores (técnicamente, se podrían dar más pequeñas, pero no se hace) limitados por máscaras intermedias entre la clase B y la C. Por ejemplo, si alguien quiere una red de un millar de direcciones necesita cuatro redes de clase C. En lugar de proporcionarle estas cuatro redes independientes, se le da una máscara de 22 bits: quedan 10 para direcciones de terminal, lo que permite 1.024 terminales.

8.1.1. Máscaras de red

Cuando un administrador de sistemas recibe el encargo de gestionar un conjunto de direcciones, es posible que necesite configurar internamente diferentes LAN con este conjunto. Por ello, el mecanismo para distinguir distintas redes (LAN) entre sí no se puede basar exclusivamente en los bits identificadores de clase que hemos comentado con anterioridad.

La máscara de red constituye el mecanismo que nos permitirá conseguir más flexibilidad. Por medio de una máscara de 32 bits, definiremos los bits que identifican la red (bits en 1) y los que identifican la estación (bits en 0). Por norma general, los bits 1 y los 0 son consecutivos, pero no necesariamente.

A continuación, definimos de nuevo el concepto *identificador de red*, adaptándolo a la máscara: el **identificador de red** es la porción de dirección IP que encaja con los bits 1 de la máscara.

El concepto *máscara* es capital para la comprensión del funcionamiento de las redes IP, permite a una estación decidir si el destino al que debe transmitir un paquete se encuentra dentro de la misma red de área local que este último o si, por el contrario, se encuentra en una LAN remota y, por tanto, debe delegar su transmisión a algún equipo de su misma LAN (el direccionador) para que se encargue de hacer llegar el paquete a su destino.

ANOTACIONES



Todas las estaciones de una misma red de área local deben utilizar el mismo identificador de red y es preciso que todas las estaciones posean la misma máscara.

Nota

Si tenemos dos estaciones con las direcciones 147.83.153.100 y 147.83.153.200, podemos deducir que están interconectadas directamente (por una LAN) si la máscara de su red es 255.255.255.0, así como deduciríamos que no están conectadas con la misma LAN si la máscara fuese, por ejemplo, 255.255.255.128.

Nota

Una notación alternativa es proporcionar el número de bits 1 de la máscara. Así pues, la máscara 255.255.255.0 es una máscara de 24 bits y la 255.255.255.128 es una máscara de 25 bits.

En ocasiones, podemos ver una dirección con el añadido de la máscara; por ejemplo:

147.83.153.100/24.

Sin embargo, esta notación sólo es útil para máscaras con 1 consecutivos.

Cuando la máscara no coincide exactamente con la estructura de clases definida, entonces se habla de *subnetting*, porque estamos creando subredes dentro de una red.

Ejemplo

La máscara 212.45.10.0/27. Permite 6 subredes distintas dentro de la red de clase C 212.45.10.0.

8.1.2. Direcciones de propósito especial

Existen diferentes direcciones especiales. Nosotros expondremos a continuación sólo las más importantes:

- **Dirección de red**. Las direcciones de red se expresan con la dirección que tendría cualquier estación suya y con todos los bits del

ANOTACIONES

identificador de estación a cero. Por ejemplo, la red en que se encuentra la estación 147.83.153.100/24 es la 147.83.153.0/24 y la red en que se encuentra la estación 147.83.153.200/25 es la 147.83.153.128/25.

- **Dirección 0.0.0.0.** Esta dirección señala al mismo ordenador que la envía. Tiene dos funciones básicas:

– Aparecer como dirección origen en paquetes IP generados por estaciones sin dirección IP asignada. Normalmente sólo aparece mientras la estación intenta averiguar su dirección mediante protocolos como RARP (*reverse address resolution protocol*), BOOTP (*bootstrap protocol*) o DHCP (*dynamic host configuration protocol*).

– Servir al software de gestión de direccionamiento para indicar la ruta por defecto.

Ejemplo

La dirección 127.03.87 es equivalente a la dirección 127.0.0.1.

- **Dirección 127.0.0.1** (***loopback***). Esta dirección no es válida para los paquetes IP. El software de red la utiliza para transmitir paquetes a la máquina local (de hecho, los paquetes no son enviados, sino que son entregados al destino por el mismo sistema operativo). En realidad, los tres bytes del identificador de estación son irrelevantes. Esta dirección sólo tiene interés para programar aplicaciones; los sistemas de red no verán nunca que ningún paquete viaje por la red con esta dirección como origen o destino.

- **Dirección 255.255.255.255** (***broadcast***). Esta dirección sólo es válida como dirección de destino de un paquete. Se utiliza para transmitir paquetes a todas las estaciones localizadas dentro de la misma LAN que la máquina de origen. Existe una versión equivalente, que es el *broadcast* dirigido. En este segundo caso, el paquete es recibido por todas las máquinas de una LAN especificada por el identificador de red. El identificador de estación debe ser todo 1.

Por lo tanto, para enviar un *broadcast* a la red 147.83.153.0 con la máscara 255.255.255.0 (o 147.83.153.0/24) podemos utilizar la dirección 255.255.255.255 si estamos dentro de la red 147.83.153.0, o bien la 147.83.153.255 si estamos en una es-

ANOTACIONES

tación remota. El primer caso, lo llamaremos *broadcast local*, y el segundo, *broadcast remoto*.

- Todas las direcciones de estos rangos:

  – 10.0.0.0/8
  – De la 172.16.0.0/16 a la 172.31.0.0/16.
  – De la 192.168.0.0/24 a la 192.168.255.0/24.

  Estas direcciones, que corresponden respectivamente a redes de clase A, B y C, no son asignadas por Internet, ni nunca lo serán. Se utilizan en redes que trabajan con los protocolos TCP/IP pero no está previsto que se conecten directamente a Internet y, en caso de que se conectaran, estarían parcialmente ocultas por *proxies* o *firewalls*, que se encargan de reducir su dirección a otra que esté en los rangos de direcciones públicas.

Por tanto, en las redes de clase A, B y C no tenemos 2^8, 2^{16} o 2^{24} estaciones posibles, respectivamente, sinó $2^8 - 2$, $2^{16} - 2$ y $2^{24} - 2$. Las direcciones que tienen todos los bits correspondientes a la estación a 0 y las que los tienen todos a 1 no son direcciones válidas para estaciones.

Actividades

- Indicad a qué clase pertenecen, cuál sería la dirección de red y la dirección de *broadcast* remoto de cada una de las direcciones IP siguientes:

  – 169.5.10.10 / 16
  – 124.127.122.123 / 8
  – 199.134.167.175 / 27
  – 201.201.202.202 / 24
  – 129.11.189.15 / 20

- Una empresa quiere montar una red de ordenadores en un entorno TCP/IP. Para ello, ha decidido usar direcciones reservadas para intranet. Concretamente la red 192.168.206.0, pero sólo necesita espacio de direcciones para diez estaciones, y se plantean hacer una segmentación de la red.

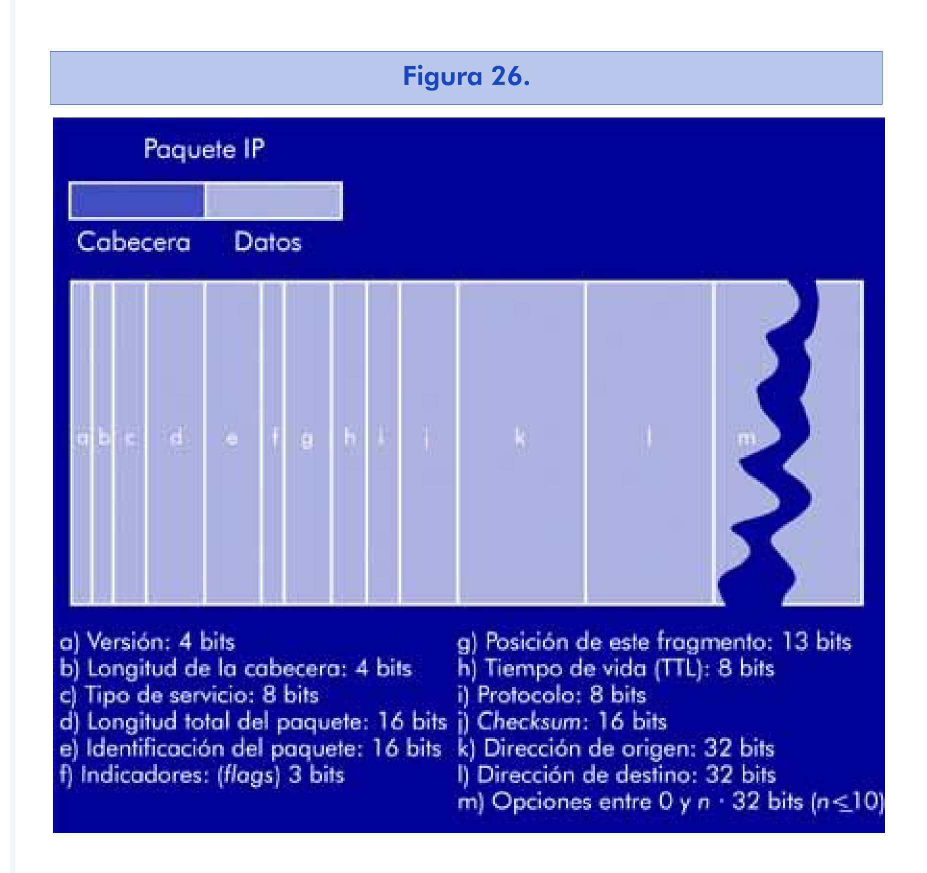
– ¿Cuál sería la máscara más restrictiva para este escenario?
– ¿Cuál sería el rango de subredes posibles?

ANOTACIONES

– Suponiendo que la dirección IP de una estación es 192.168.206.100, ¿cuáles serían las direcciones del resto de estaciones?

8.2. El formato del paquete IP

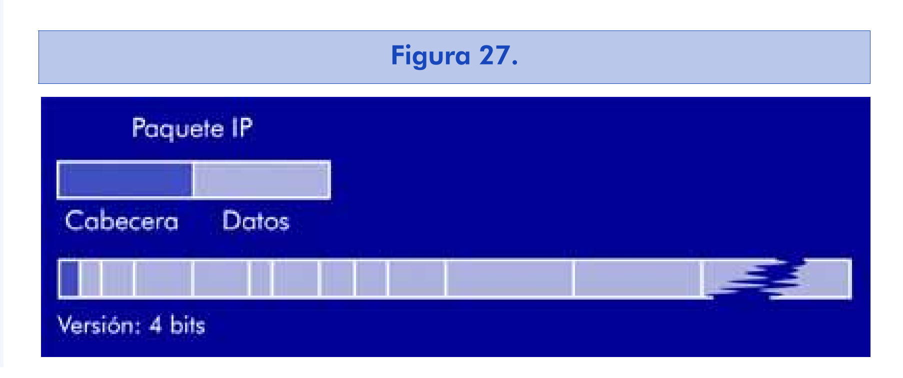
Figura 26.



A continuación describiremos los campos que componen el paquete IP:

- **Versión**: siempre vale cuatro (0100), para los paquetes de la versión actual (IPv4).

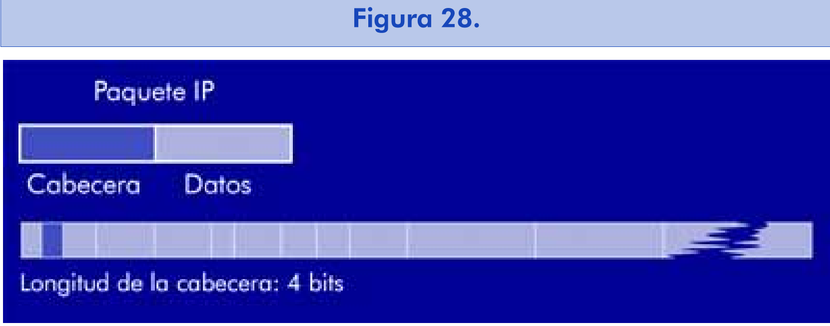
Figura 27.



ANOTACIONES

- **Longitud de la cabecera**: da la longitud de la cabecera en palabras de 32 bits (4 bytes). Por tanto, el número de bytes de la cabecera tiene que ser múltiplo de 4. Asimismo, limita la longitud de la cabecera a 60 bytes (15 · 4), puesto que 15 es el máximo que se puede expresar con cuatro dígitos binarios. Si no hay campo de opciones, la cabecera tiene 20 bytes; por tanto, el campo en este caso valdría 5 (5 · 4 bytes = 20 bytes).

Figura 28.



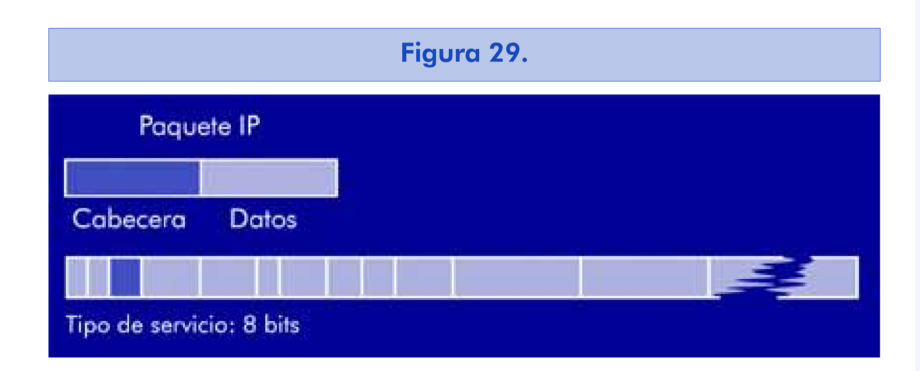
- **Tipo de servicio**: este campo se encuentra dividido en varios subcampos. Permite pedir un trato especial para el paquete y raramente se implementa.

Nota

El campo Tipo de servicio permite definir dos variables:

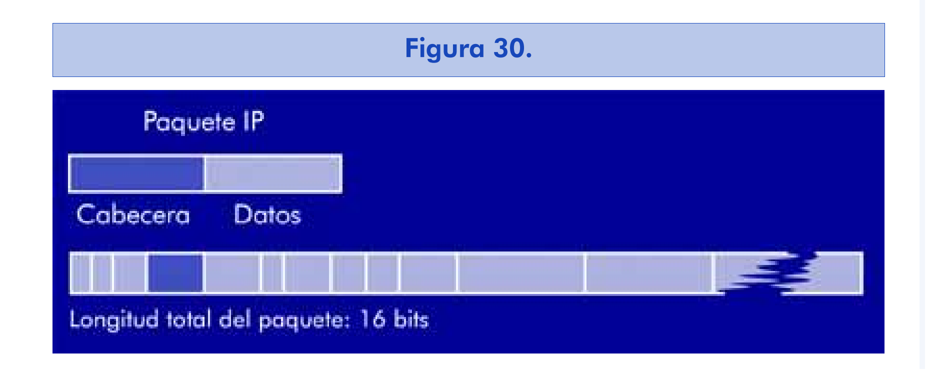
- El nivel de prioridad (del 1 al 8).
- El tipo de calidad aplicable al contenido (retardo bajo, velocidad alta, fiabilidad alta, coste bajo, etc.).

Figura 29.



- **Longitud total del paquete**: da la longitud total del paquete (cabecera incluida) en bytes. Como este campo es de 16 bits, un paquete IP no puede tener más de 65.535 bytes ($2^{16} - 1$).

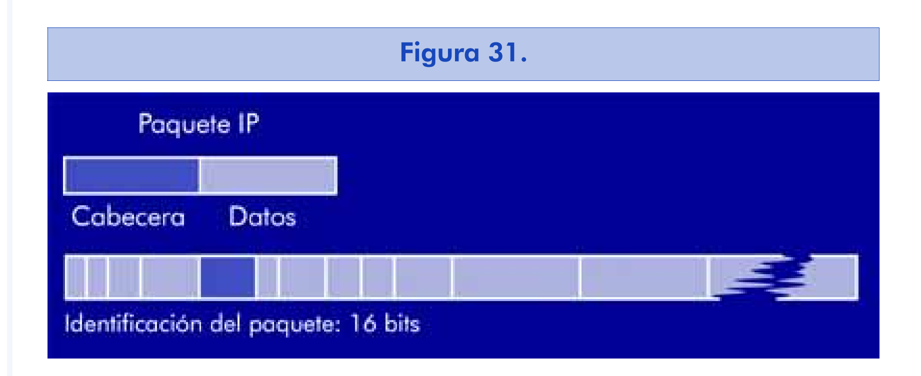
Figura 30.



ANOTACIONES

- **Identificación del paquete**: contiene un identificador que es diferente para cada paquete que genera la estación.

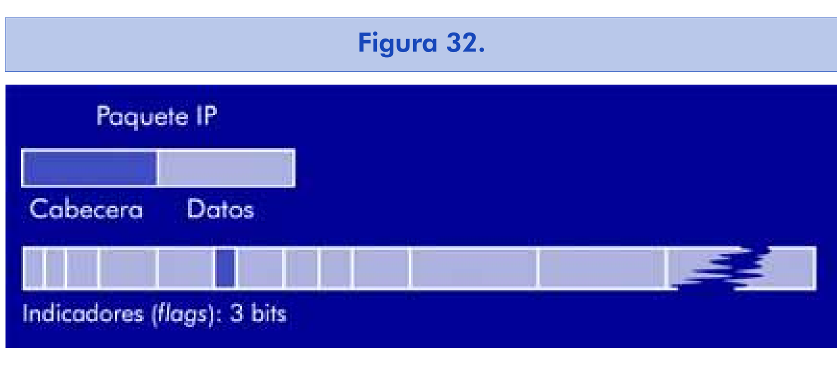
Figura 31.



> **Nota**
> Podéis ver la fragmentación de paquetes en el apartado 8.2.1 de esta unidad.

- **Indicadores** (*flags*) **y Posición de este fragmento**: estos campos permiten gestionar la fragmentación de paquetes.

Figura 32.

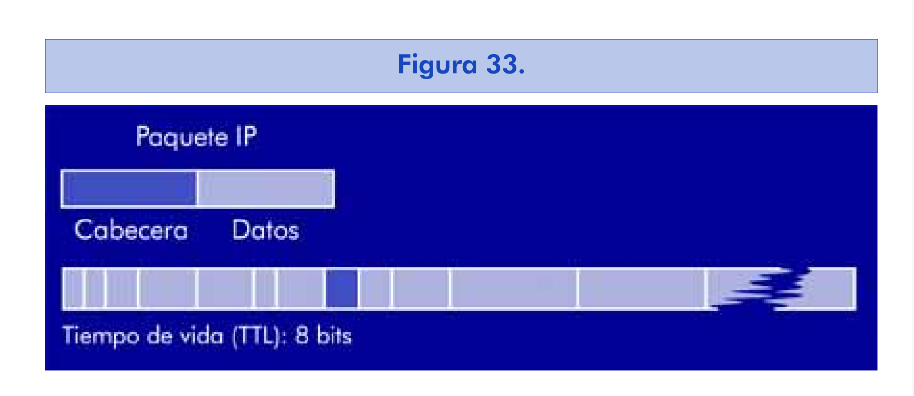




- **Tiempo de vida** o **TTL** (*Time To Live*): indica el número máximo de direccionadores que puede cruzar el paquete. Se utiliza para evitar que un paquete pueda quedar dando vueltas indefinidamente dentro de la red en caso de que haya algún problema al entregarlo. Cada direccionador disminuye el campo en uno; cuando uno de ellos detecta un paquete con TTL = 1, lo elimina y envía a quien lo ha emitido un mensaje de error por medio de un mensaje ICMP.

ANOTACIONES

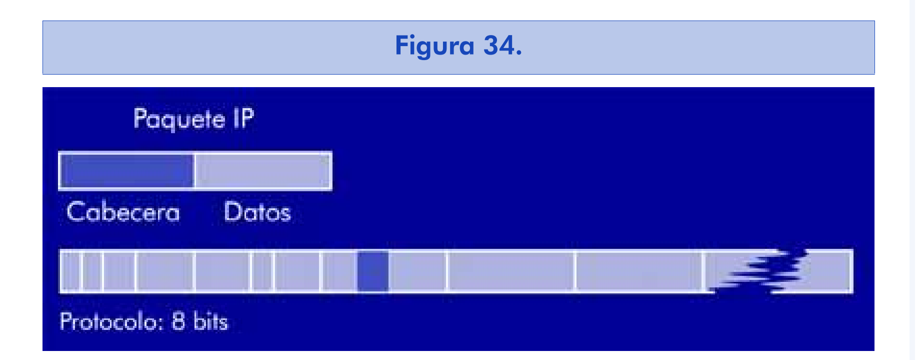
Figura 33.



- **Protocolo**: identifica el tipo de protocolo que transporta el paquete. Los valores más comunes de los diferentes tipos de protocolos son los siguientes:

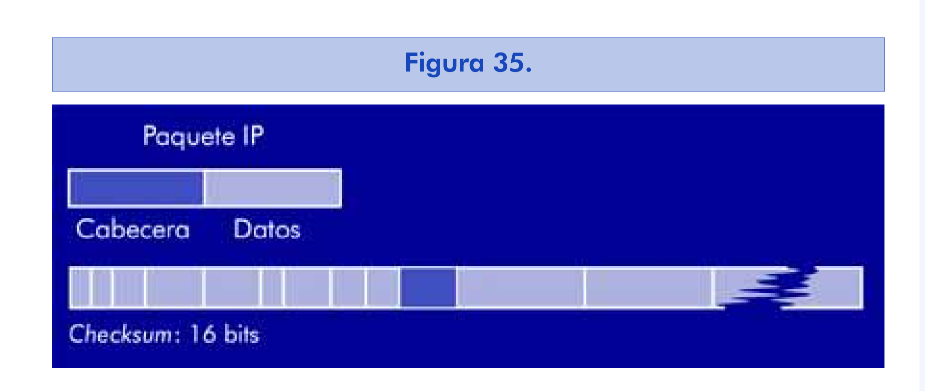
TCP	6
UDP	17
ICMP	1

Figura 34.



- *Checksum*: realiza el control de errores en la cabecera. El campo de datos no queda protegido por ningún *checksum*; es responsabilidad de los usuarios del IP (TCP, UDP, etc.) el control de los posibles errores en su contenido.

Figura 35.



Nota

Podéis ver una implementación en lenguaje C del algoritmo *checksum* en el anexo 1.

ANOTACIONES

Nota

El *checksum* (o comprobación por adición) es básicamente la suma aritmética de los bytes de la cabecera agrupados de dos en dos (si el resultado necesita más de 16 bits, se suman los bits sobrantes al mismo resultado).

Este algoritmo, si bien es fácil y rápido de calcular, no se caracteriza por poseer unas grandes cualidades para la detección de errores. Ello, sumado al hecho de que el contenido del IP no tiene *checksum* y a otros factores (tales como que muchos sistemas no calculan el *checksum* de los paquetes UDP), demuestra que en el mundo de Internet no existe un interés especial por la detección de errores. Con frecuencia, éste ha sido un argumento en el que se han basado los detractores del protocolo IP para atacarlo.

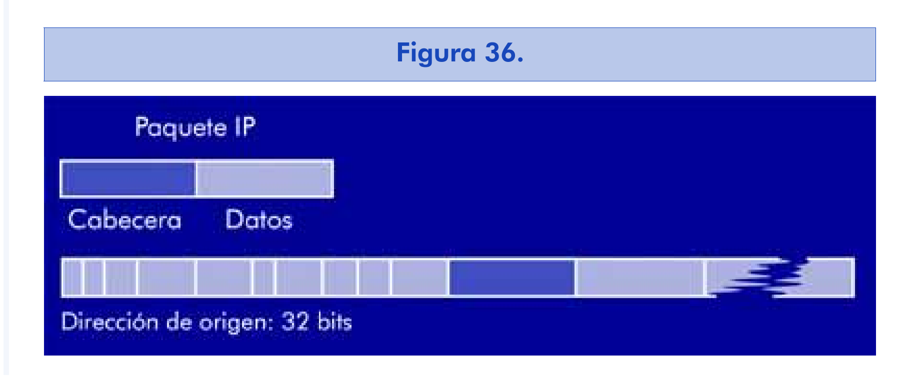
La detección de errores dentro del campo de información es responsabilidad de quien entra la información. Los usuarios más habituales del IP son los protocolos TCP, UDP e ICMP, y todos protegen la información con un campo adicional de *checksum*.

Actividad

Buscad errores posibles que pasarían inadvertidos al algoritmo *checksum*.

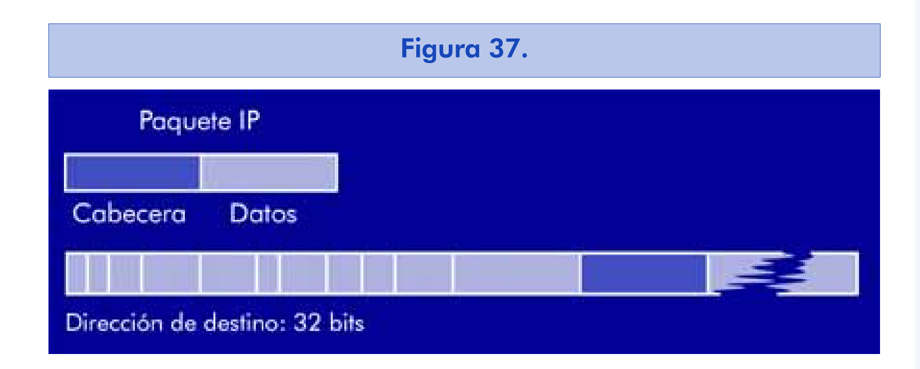
- **Dirección de origen IP**: identifica la máquina que ha generado el paquete.

Figura 36.



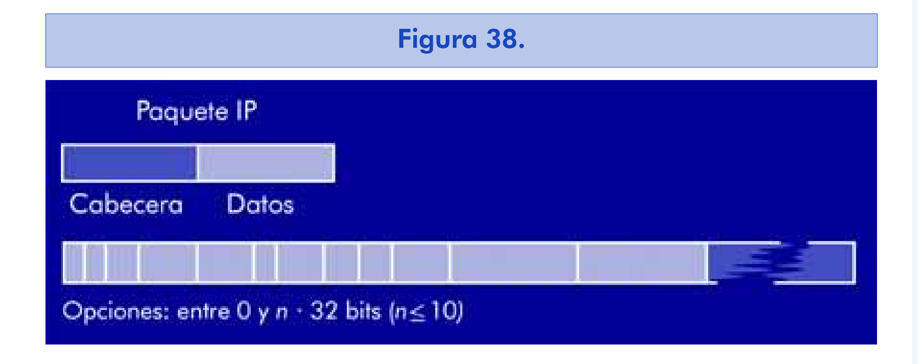
- **Dirección de destino IP**: identifica la máquina a la que va destinado el paquete.

Figura 37.



- **Opciones**: existen diferentes servicios asignados a este campo, pero por lo general no se utilizan. Comentarems algunos de los mismos cuando hablemos del ICMP. Sea como sea, la longitud total está limitada a 40 bytes (15 · 4 – 20).

Figura 38.



8.2.1. Fragmentación

Los paquetes IP van siempre insertados en tramas de nivel de enlace o MAC. Unos ejemplos que ya hemos visto, son los protocolos PPP y Ethernet.

La mayoría de protocolos de nivel de enlace fijan una longitud máxima de datos a transmitir en una trama. Esta longitud se conoce como MTU (*maximum transfer unit*) y está determinada por diferentes factores según el caso. En el caso de Ethernet, la MTU vale 1.500 bytes.

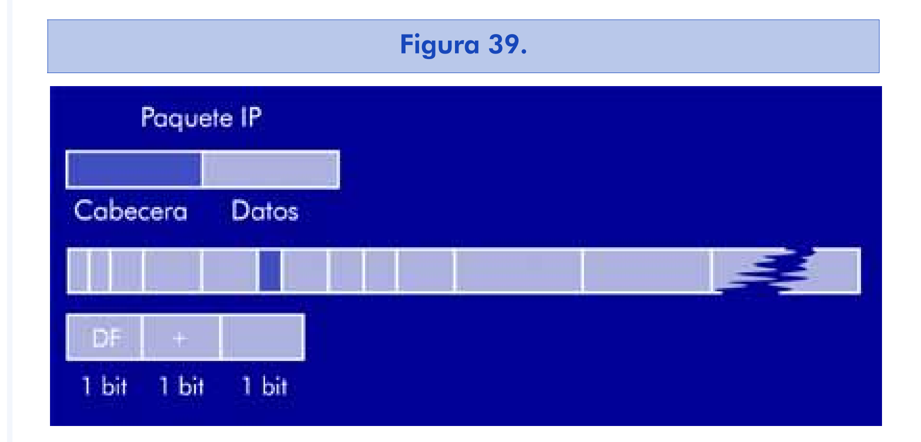
Cuando una estación transmite un paquete IP, conoce la MTU de su interfaz de red, y está restringida a transmitir paquetes IP cuya longitud sea inferior o igual a esta MTU. Si el paquete pasa por un direccionador conectado a una red con una MTU inferior al tamaño del paquete, es pre-

ANOTACIONES

ciso fragmentarlo. Los fragmentos resultantes de la operación son paquetes normales, pero con las características siguientes:

- Todos los fragmentos que provienen de un paquete fragmentado tienen el mismo identificador de paquete en las respectivas cabeceras IP.
- Todos los fragmentos indican con el campo Posición de este fragmento a qué byte corresponde el primer byte de datos del fragmento dentro del paquete original. El primer fragmento tiene un cero en este campo.
- La parte de datos de cada fragmento contiene un número de bytes múltiplo de 8. El último fragmento no tiene porque cumplir este requisito.
- El bit + del campo Indicadores es 1 en todos los fragmentos a excepción del último, que, por ello se suele denominar *hay más*.
- El resto de los campos de la cabecera se copia íntegramente del paquete original a los fragmentos que resultan de la fragmentación, excepto el indicador de longitud del paquete y el *checksum*, que se deben calcular de nuevo.

Figura 39.



En principio, podríamos suponer que, cuando los fragmentos vuelven a una red con una MTU suficiente, el direccionador que los recibe los reunifica. Sin embargo, no se hace así, en primer lugar porque la ley no escrita de Internet según la cual el direccionador debe llevar a cabo el mínimo trabajo necesario no se cumpliría. En segundo lugar (y más importante), porque nadie garantiza que todos los fragmentos sigan el mismo camino. La única estación capaz de recomponer los fragmentos es la de destino.

ANOTACIONES

La estación de destino, cuando recibe un fragmento, reserva suficiente memoria para alojar el mayor paquete IP posible (65.535 bytes), puesto que no tiene manera de saber cuál es el tamaño del paquete original. A partir de aquí, pone en marcha un temporizador* y empieza a recolectar los fragmentos según su identificador. Cuando se han recibido todos los fragmentos, se entrega el paquete a la aplicación (protocolo) correspondiente. En caso de que el temporizador salte, se descartan todos los fragmentos llegados hasta aquel momento. El nivel superior (TCP) es el responsable de pedir una retransmisión cuando convenga, puesto que el IP no dispone de ningún mecanismo para pedirla.

Nota

Este temporizador permite detectar la pérdida de un fragmento por el camino y, por tanto, la pérdida de la memoria reservada.

Nota

Dentro del campo Indicadores hay dos bits más que no hemos comentado: uno no se utiliza y el otro, el DF (*Don't Fragment*), especifica que el paquete no debe fragmentarse. Si fuera preciso fragmentarlo, el paquete sería descartado y se enviaría un mensaje ICMP indicando el error a la estación originadora del paquete. Usualmente, se activa el bit DF sólo cuando la fragmentación no es deseable como, por ejemplo, cuando enviamos paquetes a estaciones con la pila TCP/IP implementada en ROM, puesto que entonces se implementa sólo una mínima parte de la funcionalidad de TCP/IP.

Actividad

Se debe transmitir un datagrama de 4.480 bytes y necesita ser fragmentado porque pasará por una red Ethernet que admite un máximo de 1.500 bytes de datos. Indicad los valores del campo Longitud del paquete, del campo Posición de este fragmento y del bit + (MF), para cada uno de los paquetes IP que se generen.

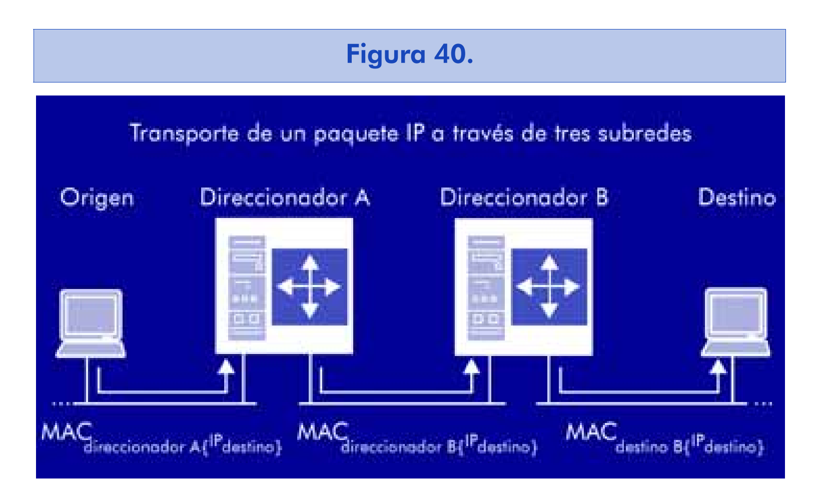
8.3. Direccionamiento y direccionadores

Las atribuciones principales de los direccionadores son interconectar dos o más subredes IP y encargarse de direccionar el tráfico destinado a estaciones remotas.

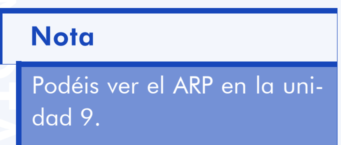
ANOTACIONES

Cada direccionador de la red Internet debe ser capaz de decidir (en una fracción de segundo) a dónde debe dirigir un paquete basándose exclusivamente en la dirección de destino, y en el conocimiento que tiene de la red y de su posición (la del direccionador) dentro de Internet. De hecho, cada direccionador no decide la ruta entera del paquete, sino sólo el trozo de ruta en que participa: el salto (*hop*) siguiente. Entre el origen y el destino tenemos un número variable de saltos, y en cada uno de ellos participan un par de direccionadores (el emisor y el receptor del paquete) a excepción del primero y el último, en que participan las estaciones emisora y receptora, respectivamente. Las conexiones entre direccionadores, o bien entre direccionador y estación terminal, las componen LAN o enlaces punto a punto.

Figura 40.



En la figura anterior, podemos ver un paquete IP que cruza tres redes de área local. En cada una el paquete va encapsulado dentro de una trama de un tipo adecuado a la subred que cruza (en la figura podrían ser tres LAN Ethernet). Las direcciones de origen y de destino del nivel LAN en cada tramo son diferentes. En cambio, la dirección de origen y la de destino de cada paquete IP no varían. La situación plantea dos problemas y, para resolverlos, es preciso efectuar los pasos siguientes:



Nota

Podéis ver el ARP en la unidad 9.

- Averiguar cuál es la correspondencia entre direcciones IP y MAC (sólo en los casos en que el enlace entre los direccionadores sea una LAN). Ello nos permitirá enviar los paquetes IP a un direccionador a pesar de que el paquete IP no tenga su dirección. El mapeado IP-MAC se efectúa de manera automática por medio del ARP.

- Decidir en cada momento cuál es el direccionador siguiente (el salto siguiente).

8.3.1. La tabla de direccionamiento

El direccionamiento se lleva a cabo a partir de **tablas de direccionamiento** que disponen de información limitada, pero suficiente para permitir la interconexión de todas las subredes que componen Internet. Cada equipo conectado a una red IP necesita una tabla de direccionamiento.

Nota

Estas tablas se introducen dentro del direccionador por medio de un terminal que se conecta al mismo directamente (consola). Asimismo, existen protocolos que permiten que las tablas se actualicen automáticamente cuando se detectan cambios de topología. Los más utilizados son el RIP (*routing information protocol*) y el OSPF (*open shortest path first*).

El direccionador decide la ruta de los paquetes consultando su tabla de direccionamiento. Las estaciones terminales también necesitan una tabla de direccionamiento, aunque sólo sea para poder descubrir si se están comunicando con una estación local de la LAN (y, por consiguiente, se pueden comunicar directamente con la misma) o si el paquete IP debe ir a una estación remota (conectada a otra LAN) y, por tanto, deben dejarlo en manos del direccionador.

Tabla de direccionamiento de una estación con una única interfaz

Empezamos estudiando la tabla de direccionamiento de una estación conectada a una LAN (dirección 147.83.153.103/24) con una única tarjeta Ethernet. Por norma general, una tabla de direccionamiento dispone también de información sobre las "cualidades" de cada una de las rutas descritas, que no aparecen en el ejemplo:

Tabla 2.

Tabla de direccionamiento de la estación				
	Dirección	Máscara	Direccionador	Interfaz
1	147.83.153.103	255.255.255.255	127.0.0.1	*Loopback*
2	127.0.0.0	255.0.0.0	127.0.0.1	*Loopback*
3	147.83.153.0	255.255.255.0	147.83.153.103	ether0
4	255.255.255.255	255.255.255.255	147.83.153.103	ether0
5	0.0.0.0	0.0.0.0	147.83.153.5	ether0

ANOTACIONES

Nota

El comando `route` permite consultar y modificar la tabla de direccionamiento de una estación. Este comando está presente, con este nombre, en la mayoría de sistemas operaitvos, incluidos los GNU/Linux (y, en general, todos los sistemas Unix).

Ésta es una tabla típica que podemos encontrar en una estación conectada a Internet. Las filas de la tabla o reglas se consultan no en el orden en que el comando `route` nos las presenta, sino en orden de máscara decreciente (en primer lugar, las entradas con 32 bits, etc.). De las entradas que forman la tabla es preciso señalar lo siguiente:

- La **primera entrada** y la **segunda** permiten transmitir paquetes IP a las direcciones 147.83.153.103 y a todas las direcciones que empiecen por 127 (fijaos en la máscara de las dos entradas). En ambos casos los paquetes se envían a la interfaz virtual *loopback* (la interfaz con que se conoce el ordenador local desde el mismo ordenador local). Ninguno de los paquetes que se direccione con alguna de estas dos reglas saldrá a la red: será cortocircuitado directamente por el sistema operativo y se entregará al proceso de destino sin que nunca llegue a ninguna tarjeta de comunicaciones.

Nota

Una interfaz es una tarjeta de conexión física a la red. Ejemplos de interfaz serían una tarjeta Ethernet (por ejemplo, la Ether0) o un puerto serie al que conectamos un módem. *Loopback*, como ya se ha comentado, es un caso especial de interfaz

Para saber qué interfaces hay disponibles en un sistema GNU/Linux, se dispone del comando `ifconfig`.

- La **tercera entrada** será adoptada por todos los paquetes destinados a la red local. El campo Direccionador es el mismo que la dirección local, lo que significa que no se delegará su transmisión a ningún direccionador.

- La **cuarta entrada** tiene una importancia relativa. Nos indica que los *broadcasts* IP se restringirán a la red local.

- La **quinta entrada** (0.0.0.0/0) permite a la estación comunicarse con estaciones remotas. Notad que la máscara no tiene ningún bit en 1. Ésta es la **ruta por defecto**. El direccionador establecido para acceder a estaciones remotas queda identificado en la tabla con la dirección 147.83.153.5.

ANOTACIONES

Toda esta información se calcula a partir de tres datos que se introducen en la configuración de red:

1) La dirección local (en el ejemplo: 147.83.153.103).
2) La máscara de la LAN local (en el ejemplo: 255.255.255.0).
3) La dirección del direccionador (en el ejemplo: 147.83.153.5).

En casos más complejos (más de una interfaz de acceso a Internet, más de un direccionador en la red local, etc.), la información deberá modificarse con el comando `route`. Asimismo, existen otros mecanismos que no requieren la intervención humana y que permiten descubrir direccionadores que no se han configurado previamente por medio del protocolo ICMP.

Actividad

Observad la tabla de direccionamiento de vuestro ordenador por medio del comando `route`. Consultad la ayuda disponible en el sistema operativo (en GNU/Linux haced `man route`).

En un direccionador, las tablas de direccionamiento tienen más entradas que las de una estación; sin embargo, el funcionamiento es el mismo que hemos visto en el caso anterior. Observad una que conecta la red 147.83.153.0/24 con el exterior por medio de la red 147.83.30.0/24:

Tabla 3.

Tabla de direccionamiento del direccionador				
	Dirección	Máscara	Direccionador	Interfaz
1	147.83.153.5	255.255.255.255	127.0.0.1	*Loopback*
2	147.83.30.2	255.255.255.255	127.0.0.1	*Loopback*
3	127.0.0.0	255.0.0.0	127.0.0.1	*Loopback*
4	147.83.153.0	255.255.255.0	147.83.153.5	ether1
5	147.83.30.0	255.255.255.0	147.83.30.2	ether0
6	255.255.255.255	255.255.255.255	147.83.153.5	ether1
7	0.0.0.0	0.0.0.0	147.83.30.1	ether0

ANOTACIONES

Prácticamente no observamos nada nuevo. Simplemente se han duplicado las entradas específicas de interfaz. El direccionador también tiene una entrada por defecto. En este caso, todo el tráfico no destinado a las dos redes a que está directamente conectado se direcciona hacia otro direccionador situado en la red 147.83.30.0 (el 147.83.30.1).

Actividad

Pensad si todos los direccionadores de Internet poseen una ruta por defecto y razonadlo. Imaginad qué sucede cuando enviamos un paquete a una estación de una red inexistente (por ejemplo, a la estación 10.0.1.1).

ANOTACIONES

9. El ARP (*address resolution protocol*)

Al describir el funcionamiento de los direccionadores hemos visto que necesitan saber la dirección MAC correspondiente a una dirección IP. El ARP es el encargado de llevar a cabo la resolución automática del mapeado entre direcciones MAC.

Cuando efectuamos la transmisión de un paquete entre dos estaciones locales de una misma LAN, lo hacemos indicando a la aplicación correspondiente sólo la dirección IP. Por ejemplo, si desde 147.83.153.103 queremos conectarnos a 147.83.153.100, haremos lo siguiente:

```
$ telnet 147.83.153.100
```

Nota

`telnet` es una aplicación de uso común para el aceso a servidores remotos. La veremos en la unidad 17.

Aplicando la máscara de red a la dirección IP solicitada, la estación deduce que se encuentra en su misma subred. Por tanto, no hace falta delegar en ningún direccionador. Para poder enviarle las tramas en las que deben ir los paquetes IP que genera la aplicación `telnet`, necesita conocer su dirección MAC.

Para ello, emite una petición ARP. Se trata de un paquete encapsulado directamente sobre una trama Ethernet (con tipo = 0x0806). Como dirección de destino lleva la dirección broadcast (FF:FF:FF:FF:FF:FF) para que llegue a todas las estaciones de la LAN, y como contenido, la dirección IP para la cuál se desa conocer la dirección MAC. La estación que reconoce su IP en la petición ARP responde con una respuesta ARP dirigida al origen de la petición con su dirección MAC. De hecho, el contenido de la respuesta ARP es irrelevante, lo único importante es la dirección MAC de origen de la trama.

Nota

No mostraremos el formato de los paquetes ARP, porque no van a aportarnos mucha más información. Podéis encontrar el formato y los usos alternativos del ARP en:

D. E. Comer (1995). *Internet-working with TCP/IP* (volumen 1: "Principles, Protocols, and Architecture"). Hertfordshire: Prentice Hall

Lo mismo sucede cuando la estación deduce que la IP solicitada por alguna aplicación corresponde a una estación remota, o sea, de fuera de su LAN. En este caso, las tramas se deben enviar al direccionador para que se encargue de sacarlas de la LAN hacia su destino. Para ello, la estación origen debe averiguar la dirección MAC del direccionador.

ANOTACIONES

Nota

En rigor, para cada trama con paquete IP a generar, se tendría que hacer una petición ARP.

Es fácil deducir que,en el funcionamiento normal de una estación,. las peticiones ARP pueden ser muy habituales. Para evitarlo, cada estación dispone de una tabla las parejas ya obtenidas, de forma que si una dirección IP ya se encuentra es dicha tabla, no hace falta enviar ninguna petición ARP. Dicha tabla se denomina ***caché ARP***, y tiene el siguiente aspecto:

Tabla 4.

Tabla caché ARP			
	Dirección IP	Dirección MAC	Interfaz
1	147.83.153.103	08:00:00:10:97:00	ether0
2	147.83.153.5	00:0c:aa:00:0f:e0	ether0

Cada fila corresponde a un mapeo IP-MAC y la interfaz para la cual se ha solicitado.

Nota

En GNU/Linux, `arp -a` sirve para mostrar la tabla. Podéis consultar las páginas del manual para el resto de opciones.

Esta tabla se va llenando automáticamente a medida que se realizan nuevas peticiones, pero también se puede manipular con el comando `arp`. Con él se puede consultar, añadir o borrar entradas.

La tabla ARP se denomina *caché* porque, de hecho, actúa como una memoria auxiliar que evita la consulta de la información a la LAN mientras se tenga una copia local de la misma. Puede parecer que la consulta ARP no debería ser demasiado frecuente, puesto que al cabo de cierto tiempo toda la información se encontraría dentro de la caché. Conviene saber, sin embargo, que las entradas caducan al cabo de un periodo de tiempo relativamente breve (entre uno y unos cuantos minutos, según el sistema).

Actividad

Imaginad qué podría suceder si las entradas de la caché ARP no caducaran. Os puede servir de ayuda pensar en la reconfiguración de direcciones.

Algunos usos alternativos de interés del ARP son los siguientes:

- **ARP innecesario** (*gratuitous-ARP*): se utiliza cuando una estación arranca para saber si hay alguna otra estación que está utilizando su misma dirección IP. Por medio de una petición ARP, puede pre-

ANOTACIONES

guntar quién tiene su dirección IP (un conflicto de este tipo podría dejar las dos estaciones "fuera de combate").

- **ARP subsidiario** (*proxy-ARP*): se utiliza en situaciones en las que un direccionador divide una subred sin que las estaciones ni el direccionador que los conecta a Internet modifiquen su máscara. Esta técnica, a pesar de no ser demasiado ortodoxa, se aplica con frecuencia cuando el direccionador de una red privada se conecta a Internet por medio de una red ajena (un proveedor de Internet, por ejemplo).

Actividad

Consultad la caché ARP de algún sistema que tengáis al alcance (Unix o MS) por medio de `arp -a`. Comprobad qué opciones tiene disponibles.

ANOTACIONES

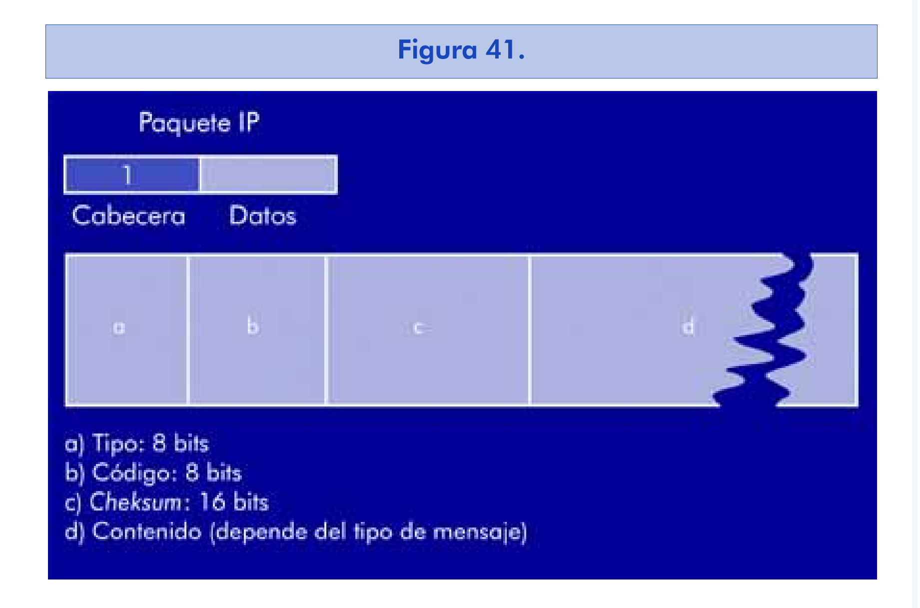
10. El ICMP (*Internet control message protocol*)

La cuestión de si el ICMP es un protocolo, o si más bien es una herramienta que utiliza el protocolo IP para notificar errores genera mucha polémica. Lo cierto es que el **ICMP** constituye el mecanismo básico para la gestión de las diferentes incidencias que pueden ocurrir en una red IP.

10.1. Mensajes ICMP

Los mensajes ICMP viajan dentro de paquetes IP (al contrario de lo que sucedía con los paquetes ARP), en el campo de datos con el campo Protocolo igual a 1. El formato del mensaje presentado en la figura siguiente nos facilitará el estudio de los diferentes usos del paquete ICMP:

Figura 41.



Existen trece *tipos* de mensajes ICMP en uso en la actualidad, y alrededor de una treintena de subtipos identificados con el campo Código.

ANOTACIONES

Tabla 5.			
Tipo	**Código**	**Descripción**	**Clase**
0 8	0 0	Respuesta de eco (*echo reply*) Petición de eco (*echo request*)	Petición Petición
3	0-15	Destino inalcanzable (*unreachable destination*) Los diferentes códigos permiten definir si lo que no se puede alcanzar es la subred (0, 6, 9, 11), la estación (1, 7, 10, 12), el protocolo (2), o el puerto (3) y los motivos	Error
4	0	Petición de control de flujo (*source quench*)	Error
5	0-3	Redireccionamiento	Error
9	0	Publicación de rutas	Petición
10	0	Petición de rutas	Petición
11	0-1	El tiempo de vida ha expirado (*time exceeded*)	Error
12	0-1	Cabecera IP incorrecta	Error
13 14	0 0	Petición de hora Respuesta de hora (en milisegundos desde la medianoche)	Petición Petición
17 18	0 0	Petición de la máscara de la subred Respuesta de la máscara de la subred	Petición Petición

La última columna nos permite distinguir mensajes ICMP de notificación de error de mensajes que son parte de una petición (la petición o la respuesta). Esta distinción es importante, puesto que los mensajes de error ICMP no pueden generar otros mensajes de error ICMP. En particular, no se generan mensajes de error en respuesta a los paquetes o mensajes siguientes:

- Los mensajes de error ICMP.
- Un paquete IP destinado a una dirección *broadcast* (sea un *broadcast* IP o un *broadcast* MAC).
- Un fragmento que no sea el primero.
- Una dirección de origen que no identifique una única estación. Por ejemplo, la dirección de origen válida 0.0.0.0 o la dirección de origen no válida 255.255.255.255.

ANOTACIONES



En cualquiera de las situaciones que acabamos de describir se pueden provocar respuestas en cascada que podrían afectar gravemente a la red.

Nota

Parches

Desgraciadamente, no todas las implementaciones TCP/IP han tenido en cuenta las excepciones existentes en la generación de mensajes de error, y algunos usuarios desaprensivos han explotado estos problemas para bloquear sistemas y redes remotas. Asimismo, de vez en cuando se descubren nuevas excepciones que pueden afectar a nuestros sistemas.

Los fabricantes de sistemas operativos publican regularmente parches (patch) que permiten solucionar los problemas (las vulnerabilidades) que se han descubierto desde la fecha de publicación de la última versión.

10.2. El programa `ping`

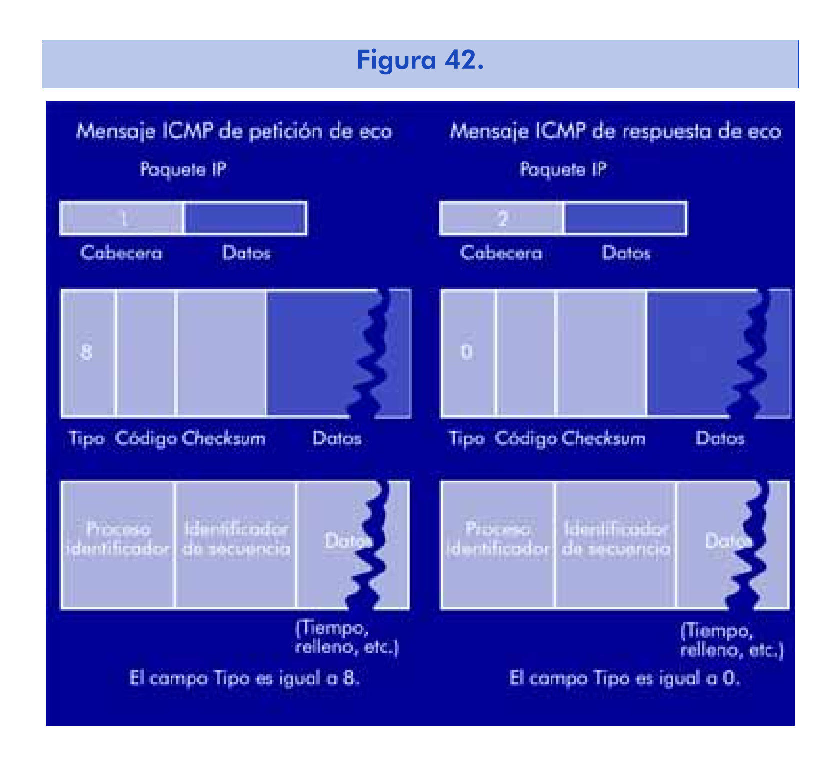
El programa `ping` permite descubrir si una estación se encuentra activa o no, simplemente efectuando lo siguiente:

```
$ ping <direccion_IP_destino>
```

El programa `ping` envía un mensaje ICMP del tipo 8 (petición de eco) con el destino indicado. El receptor de la petición debe responder con una respuesta de eco (ICMP tipo 0), y, cuando el `ping` la recibe, indica en pantalla que la estación remota está activa. Evidentemente, el programa debería llamarse ping-pong.

ANOTACIONES

Figura 42.



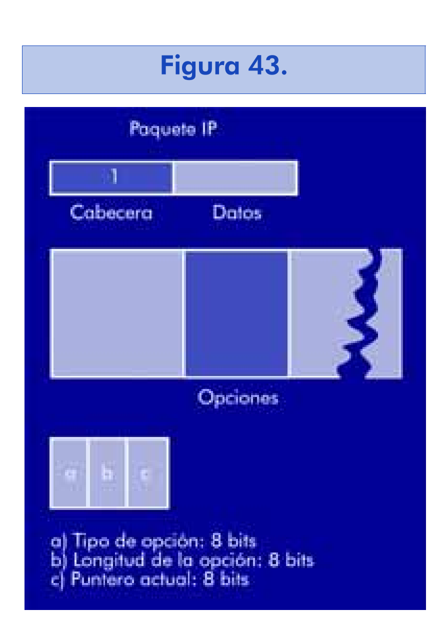
Actividad

Practicad el uso del programa `ping`. Comprobad qué retardos máximos se dan en Internet. Hacedlo a diferentes horas del día.

Con el `ping` tenemos otras opciones disponibles. En particular, la opción de memorización de rutas (*record route* o RR; `ping -R`, en GNU/Linux), que no se refleja en ninguno de los campos del mensaje ICMP, sino que se encuentra dentro de la misma cabecera IP, en el campo de opciones.

Las diferentes opciones que se encuentran disponibles en la cabecera se identifican por medio del primer byte del campo de opciones de la cabecera IP (figura 43).

Figura 43.



Si el originador, por ejemplo, quiere activar la opción RR, el primer byte debe ser 7. Sea como sea, en todas las opciones el primer byte indica el tipo, y el segundo, la longitud en bytes de la opción.

ANOTACIONES

En este caso siempre se pide el máximo posible, que es 39 bytes, puesto que el campo siguiente tiene un byte (puntero), al que sigue una cadena de campos de 4 bytes (que son las direcciones IP encontradas). El campo Puntero se inicializa a 4, y dentro de los 4 bytes siguientes se guarda la dirección de la estación en que ejecutamos el `ping`.

Cada direccionador debe mirar dentro de los paquetes IP (ICMP o no) para ver si tienen la opción RR activada. Si un direccionador encuentra un paquete con esta opción activada, modifica la posición apuntada por el puntero con su dirección (por norma general, la dirección de salida del direccionador) e incrementa el valor del puntero en cuatro unidades. Cuando vuelve el mensaje de respuesta de eco, se ha añadido una lista con todos los saltos que ha tenido que realizar el paquete (tanto de ida, como de vuelta).

El campo de opciones tiene limitaciones de tamaño: sólo dispone de 36 bytes (39 – 3) para guardar direcciones IP. Como cada dirección ocupa 4 bytes, sólo hay espacio para nueve direcciones IP. Si a ello le añadimos que no todos los direccionadores comprueban si hay opciones dentro de los paquetes IP, o no actualizan la opción RR, hace poco útil este tipo de ping en el mundo real.

Actividad

Comprobad la ruta a diferentes destinos por medio de un ping con la opción de memorización de rutas. Valorad si la comprobación de esta opción está muy extendida.

10.3. El programa `traceroute`

El programa `traceroute` permite encontrar las rutas entre un origen y un destino sin ninguna de las limitaciones del `ping -R`. Utiliza un mecanismo bastante ingenioso basado en mensajes genéricos ICMP (y no los específicos petición de eco y respuesta de eco del `ping`).

ANOTACIONES

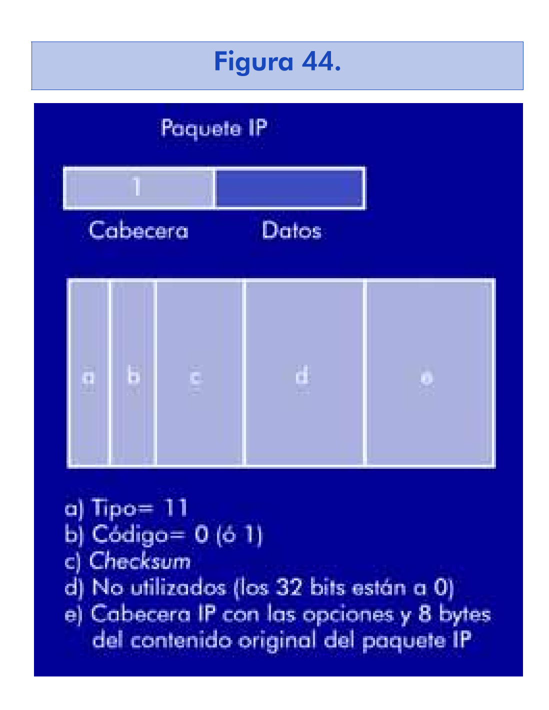
El funcionamiento se basa en la explotación de dos mensajes ICMP:

1) **Tiempo de vida agotado** (*time-exceeded*): cuando un direccionador recibe un paquete, aparte de las tareas primordiales de direccionamiento, debe reducir en una unidad el valor del campo TTL de la cabecera IP.

   En caso de que el valor (después de la reducción) sea cero, el paquete debe eliminarse. Sin embargo, esta eliminación no es silenciosa, sino que el direccionador responsable envía una notificación de la misma al originador del paquete por medio de un mensaje ICMP tipo 11 y código 0 (tiempo de vida agotado).

   Este paquete ICMP contiene la cabecera del paquete IP que se ha eliminado y los primeros 8 bytes de su contenido (seguramente la cabecera UDP o los primeros bytes de la cabecera TCP).

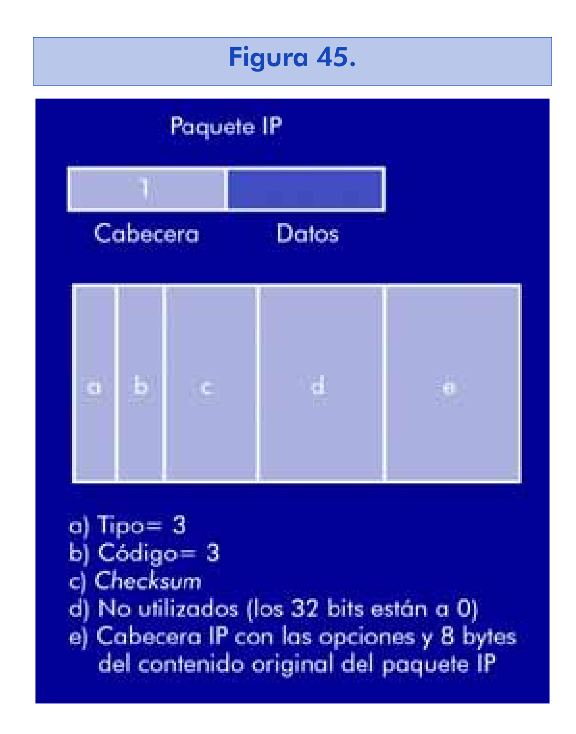
Figura 44.



2) **Puerto inalcanzable** (*unreachable-port*): cuando una estación recibe un datagrama UDP o un segmento TCP destinado a un puerto que la máquina no escucha, responde con un mensaje de error de puerto inalcanzable (tipo 3 con código 3).

ANOTACIONES

Figura 45.



El programa `traceroute` simplemente debe enviar paquetes al destino con TTL secuencialmente ascendentes: el paquete (con independencia del tipo que sea) que tenga el TTL = 1 será rechazado por el primer direccionador, el que tenga TTL = 2 lo será por el segundo, y así consecutivamente. Cada uno de los direccionadores devolverá un mensaje ICMP "tiempo de vida agotado", una pista del todo suficiente para que el originador averigüe el camino que han seguido todos los paquetes.

Cuando el mensaje llega al destino, debe devolver algún mensaje para saber que la secuencia ha finalizado. Por norma general, el mensaje será "puerto inalcanzable" si el mensaje enviado era un datagrama UDP a un puerto no usado, o bien una respuesta de eco si lo que se ha enviado son paquetes ICMP de petición de eco.

Actividades

Utilizad el programa `traceroute` para descubrir los caminos que siguen diversos paquetes hasta diferentes destinos.

Probad qué sucede si nos conectamos a un puerto no servido. Por ejemplo, conectaos al puerto TCP 1234 (podéis hacerlo con `Telnet localhost` 1234).

ANOTACIONES

10.4. Mensaje de redireccionamiento

Es normal que los sistemas se conecten a Internet sólo configurando su dirección IP, la máscara de la LAN y el direccionador que gestiona la comunicación remota. Cuando hay más de un direccionador en la LAN local, puede darse el caso de que para algunas rutas sea mejor usar otro direccionador y no el que tenemos configurado.

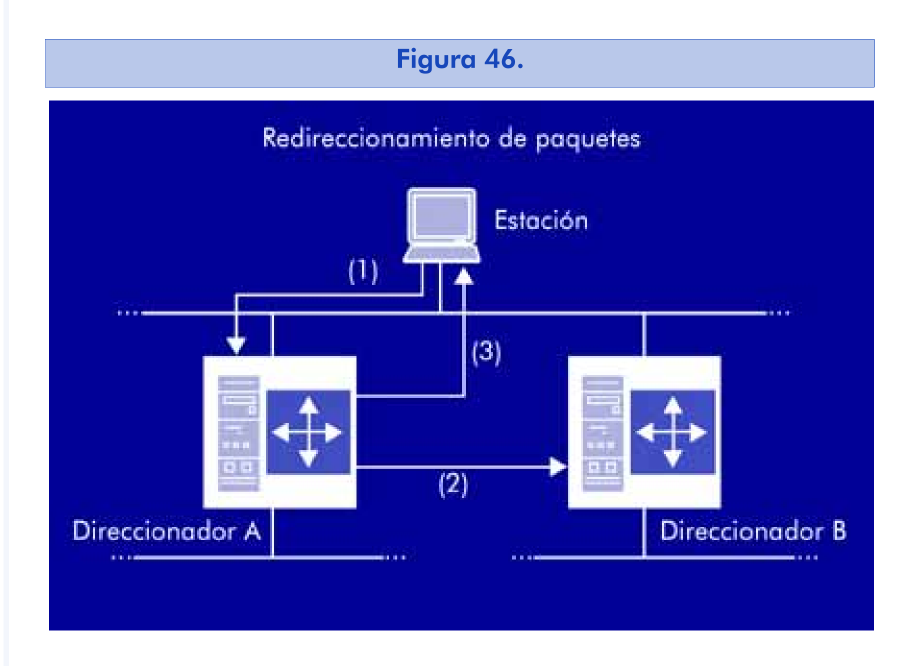
A este efecto, los direccionadores disponen del mensaje ICMP de redireccionamiento (*redirect*), que actúa de la manera siguiente:

1) La estación envía un paquete al direccionador que tiene configurado (A). El direccionador A descubre que la mejor ruta pasa por utilizar el direccionador B.

2) El direccionador A direcciona el paquete hacia el direccionador B.

3) Notifica a la estación que modifique su tabla de direccionamiento.

Figura 46.



Notad que, normalmente, el direccionador continúa direccionando los paquetes (paso 2). Aunque la estación no hiciera caso del mensaje ICMP de redireccionamiento (paso 3), continuaría teniendo conectividad con el exterior; sin embargo, si le hace caso, mejorará el rendimiento del sistema.

ANOTACIONES

Cuando una estación obedece a un ICMP de redireccionamiento, su tabla de direccionamiento queda convenientemente actualizada. El comando `route` puede proporcionarnos alguna pista de si este fenómeno tiene lugar.

ANOTACIONES

11. Redes de acceso a Internet

Las redes de acceso a Internet más habituales son la red telefónica (por módem) que se utiliza, sobre todo, en el ámbito doméstico, el ADSL (*asymetric digital subscriber line*, línea de abonado digital doméstica) que, aunque utiliza la infraestructura de acceso de la red telefónica, no se puede decir que vaya sobre la línea telefónica, y la Ethernet.

1) En accesos por medio de la **red telefónica** y, en general, en accesos por medio de redes conmutadas (incluyendo el acceso por la RDSI), se suele utilizar el PPP (*point-to-point-protocol*).

> ***Nota***
>
> Si bien durante mucho tiempo se han utilizado el SLIP (*serial line Internet protocol*) y el (*compressed SLIP*), en la actualidad se han dejado prácticamente de lado en favor del PPP, que tiene más flexibilidad (permite gestionar automáticamente ciertos parámetros IP y multiplexar, dentro de la misma conexión, diferentes protocolos de interconexión, aparte del IP) y es más fiable (dispone de CRC en cada trama).

2) En **LAN**, el protocolo que se utiliza en más del 90% de los casos es la Ethernet. Nosotros nos centraremos sólo en los detalles de las direcciones de dicho protocolo, puesto que es lo único que afecta a la manera de funcionar del IP.

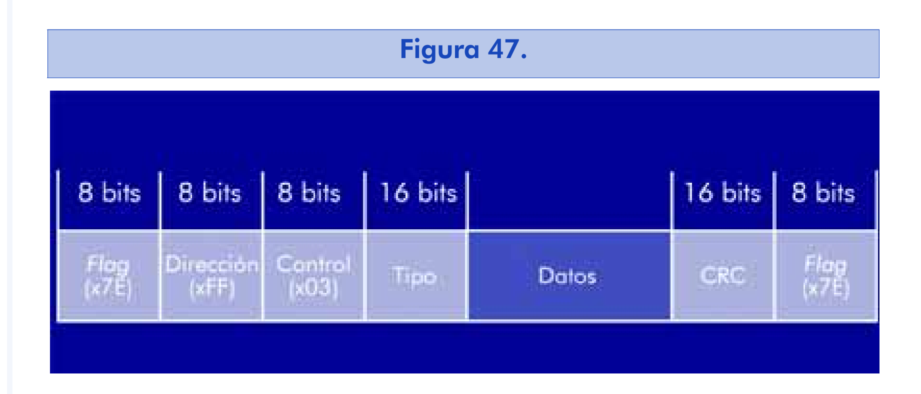
   Casi todos los protocolos de LAN que componen el 10% restante (IEEE802.3 CSMA/CD, IEEE802.5 Token Ring, etc.) utilizan una estructura de direcciones idéntica a la de Ethernet. De hecho, podríamos hablar de compatibilidad, puesto que la asignación de direcciones se hace globalmente para todas las LAN mencionadas y la gestiona el IEEE (Institute of Electric and Electronic Engineers).

ANOTACIONES

11.1. Acceso telefónico: el PPP

El PPP es fundamentalmente un protocolo derivado del HDLC (*high-level data link protocol*) para conexiones balanceadas (HDLC-ABM, *HDLC-asynchronous balanced mode*). El formato de la trama PPP se representa en la figura siguiente:

Figura 47.



Los campos Indicador (*flag*), Dirección y Control están fijados en los valores de la figura anterior. El campo Dirección tiene el valor 11111111, que es el de difusión o *broadcast* en la mayoría de los protocolos (por ejemplo, en todos los HDLC). Ello significa que este campo (como el de control) no se utiliza. Su utilización en el PPP sólo se puede justificar por el posible uso de tarjetas HDLC genéricas para conexiones PPP. Como cualquier protocolo HDLC, debe aplicar el mecanismo de transparencia de inserción de ceros (*bit stuffing*).

Nota

El PPP especifica una variante orientada a carácter (los protocolos de la familia HDLC están orientados a bit), que es la que más se utiliza en enlaces por medio de módem (un contraejemplo serían las conexiones mediante la RDSI).

Lo importante es lo que transportan las tramas PPP. El mismo estándar define la multiplexación de diferentes protocolos, que se distinguirán por medio del campo Tipo. Los que nos interesan son los siguientes:

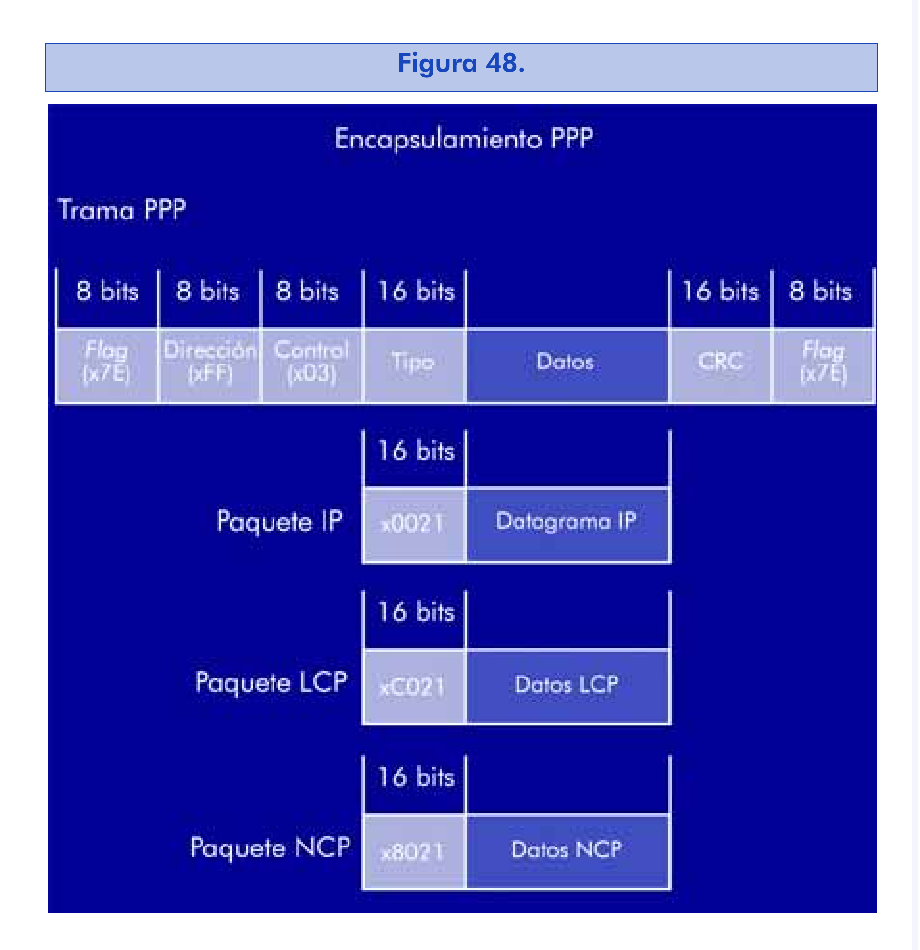
- **LCP** (*link control protocol*): es el protocolo encargado de realizar el test del enlace, el control de la conexión y la gestión del enlace.

ANOTACIONES

- **Protocolos de red**: son tramas que encapsulan paquetes de nivel superior, como puede ser IP. Pero también pueden transportar NETBEUI, AppleTalk, Decnet, etc.

- **NCP** (*network control protocol*): es el protocolo que se utiliza para tareas de gestión de los protocolos de red que transporta el enlace. En el caso del IP, permite que uno de los terminales asigne la dirección de Internet al otro y configure los diferentes parámetros de la red (direccionador, máscara, etc.)

Figura 48.



11.1.1. Compresión de las cabeceras

Como ya hemos visto, la pirámide de PDU provoca ineficiencias en la transmisión, sobre todo en protocolos como Telnet, que con frecuencia envían bloques muy cortos de información. Asimismo, el PPP se ha utilizado en enlaces telefónicos que funcionan a velocidades máximas teóricas de entre 9.600 bps (módems norma V.32) y 33.600 bps (módems norma V.34). Ello hace que dicha ineficiencia sea todavía más grave.

ANOTACIONES

Nota

Si se utiliza Telnet, para enviar un carácter (de 1 byte) debemos enviar 8 + 20 + 20 + 1 = 49 bytes. Con un módem de 9.600 bps, si utilizáramos mensajes de 8 bits de longitud más 1 bit de arranque y 1 bit de parada, podríamos transmitir:

9.600 / (1 + 8 + 1) = 960 caracteres/segundo.

De hecho, nadie es capaz de mecanografiar tan rápido. Sin embargo, no debemos perder de vista que el enlace puede ser compartido por otras conexiones (transferencia de ficheros, consulta de la web, etc.).

La **compresión de cabeceras Van Jacobson** mejora considerablemente este problema. Este tipo de compresión se basa en el hecho de que en un acceso a Internet con PPP, en general no habrá demasiadas conexiones TCP simultáneas. La compresión Van Jacobson permite mantener una tabla de hasta dieciséis conexiones simultáneas TCP/IP, a las que asignará dieciséis identificadores diferentes. Como la mayoría de los campos de las dos cabeceras TCP/IP no varían durante el transcurso de una misma conexión, basta con tener entre 3 y 5 bytes de cabecera para cada paquete (combinando PPP y TCP/IP) para mantener un funcionamiento correcto de la conexión. La ganancia en eficiencia de la transmisión es suficientemente importante.

Para saber con exactitud qué ganancia se consigue, precisamos saber qué longitud pueden tener las tramas PPP.

Por medio del LCP también pueden obtenerse otras mejoras como, por ejemplo, la eliminación de los campos de control y de dirección.

11.1.2. MTU

Como ya hemos comentado, la mayoría de protocolos de nivel de enlace poseen una longitud máxima de transmisión, la MTU, que condiciona el tamaño de los paquetes de nivel superior que tansporta. En el PPP, protocolo derivado del HDLC, no se trata de un valor concreto, sino que el límite vendrá fijado por la probabilidad de error en un bit que tengamos: cuanto más larga sea la trama, más probabilidad existe de que sea errónea.

ANOTACIONES

Sin embargo, existe un factor que limitará la MTU más que los límites impuestos por los protocolos: el **confort de la conexión**. Se ha demostrado que, en conexiones en tiempo real (como una conexión Telnet), el usuario debe recibir una reacción a sus acciones en una décima de segundo como máximo. retardos superiores provocan cansancio y dan la sensación de que "la máquina va lenta", que todos hemos experimentado alguna vez.

Si pensamos que dentro de la misma conexión podemos tener multiplexadas otras de transferencia (como FTP o web), nos encontraremos que los paquetes de las aplicaciones en tiempo real, que suelen ser cortos, deben esperarse detrás de los paquetes de longitud máxima que se utilizan en las aplicaciones que tienen una tasa de transmisión elevada (supondremos que los paquetes de la aplicación en tiempo real tienen preferencia sobre otros que previamente estuvieran en la cola de salida).

La única manera de hacer que una aplicación estándar de transferencia utilice paquetes más pequeños es reducir la MTU de la conexión. En el caso del PPP, si tomamos como referencia un módem de 33.600 bps, tenemos que un paquete de una décima de segundo de duración tendría los bytes siguientes:

$$33.600 \text{ bps} \cdot (0{,}1 \text{ s}) \cdot (1 \text{ byte} / 8 \text{ bits}) = 420 \text{ bytes.}$$

En conexiones PPP tendremos, pues, la MTU entre 250 y 500 bytes. Con módems que dispongan de compresión, los valores que obtendremos pueden aumentar.

Nota

Las velocidades de transmisión de los módems estándar son 9.600 bps (V.32), 14.400 bps (V.32bis) y 33.600 bps (V.34).

En general, los módems disponen de dispositivos electrónicos para comprimir y descomprimir datos. Los estándares V.42bis y MNP9 realizan compresiones de hasta 1:4, con lo que pueden lograr velocidades de transmisión efectiva de hasta 134.400 bps (V.34 + V.42bis) en situaciones favorables.

ANOTACIONES

Hay módems (los que siguen el estándar V.90) que pueden lograr velocidades de hasta 56 kbps. En realidad, no son módems en el sentido estricto, puesto que necesitan que en el otro extremo de la línea haya un codec conectado a una línea digital.

11.2. Acceso ADSL

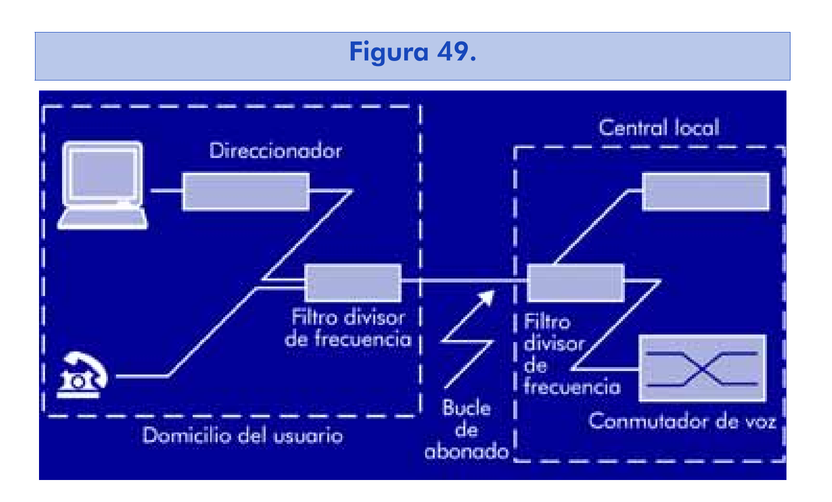
Si bien el acceso a Internet por módem por medio de PPP ha sido la manera habitual de conexión por parte de los usuarios domésticos y las pequeñas empresas durante la primera "década Internet", parece que ello cambiará y en la segunda "década Internet" se adoptarán sistemas que faciliten la conexión permanente.

Nota

La conexión por módem ocupa la línea de teléfono del abonado y, lo que todavía es peor, establece una llamada durante un tiempo indefinido.

El ADSL representa una solución a este problema, puesto que, por un lado (aunque utiliza el cableado de la línea telefónica, el bucle de abonado), la línea telefónica queda libre para llamadas mientras se está conectado a Internet y, por otro, la conexión no consume recursos de la red telefónica, puesto que, cuando la señal llega a la centralita, se extrae del bucle de abonado y pasa a la red IP de la operadora.

Figura 49.

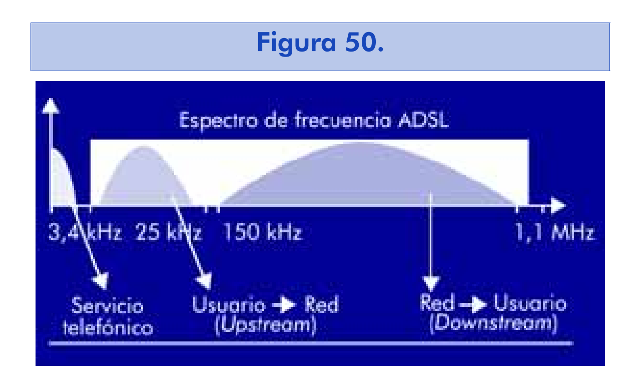


ANOTACIONES

En general, el direccionador que hay en el domicilio del usuario ofrece una conexión Ethernet equivalente a un concentrador, lo que permite que el abonado conecte más de un ordenador por medio de la línea ADSL.

Para permitir que la línea telefónica conviva con la conexión a Internet permanente, se realiza una división del espectro: la parte baja continúa ocupada por el canal de voz (hasta 4 kHz) y, a partir de aquí, se sitúa el espectro de la codificación ADSL (con la limitación propia del par de hilos). El sistema es bidireccional: reserva el espectro bajo para la salida a la red, y el resto, para la entrada:

Figura 50.



Generalmente, no se puede llegar a los 1,1 MHz que indica la figura, puesto que la calidad del par de hilos es muy variable (clima, longitud, edad, etc.), por lo que se utiliza una codificación adaptativa: los dos sentidos del canal se dividen en subbandas de 4 kHz independientes (DMT, *discrete multitone*) que se codifican (en realidad, se modulan) con procedimientos casi idénticos a los utilizados por los módems tradicionales. Con ello, se consiguen treinta y dos canales de salida y hasta doscientos cintuenta y seis de entrada, con una capacidad de 60 kbps cada uno de los mismos –modulados en QAM (*quadrature amplitude modulation*), como los módems telefónicos– que, acumulados, proporcionan un máximo de 15,36 Mbps de entrada y 1,92 Mbps de salida.

No obstante, las mejores conexiones no consiguen aprovechar correctamente los canales superiores y se considera que el límite máximo de salida es de 8 Mbps. Asimismo, las operadoras no suelen ofrecer conexiones tan rápidas, lo que hace que no se pueda lograr el límite teórico.

Los protocolos utilizados dentro del ADSL dependen de la operadora y no están definidos por el estándar. Entre los posibles protocolos para el ADSL, tenemos el ATM y el mismo PPP.

Nota

Algunas operadores, para aprovechar mejor líneas malas, utilizan una variante del estándar que permite la transmisión bidireccional con los dos canales en la misma banda.

Esta solución, aunque necesita DCE más caros, permite aprovechar líneas con peores condiciones y/u obtener velocidades más altas.

ANOTACIONES

11.3. Acceso LAN: el protocolo Ethernet

Seguramente, la simplicidad de este protocolo de red, y no sus prestaciones teóricas, ha hecho que sea el más utilizado en redes de área local prácticamente desde que DEC, Intel y Xerox establecieron un estándar (*de facto*) para LAN con control de acceso al medio CSMA/CD, basándose en una arquitectura similar desarrollada los años setenta por Xerox.



Hay un principio que se cumple en todas las **redes de área local**: lo que una estación transmite es recibido por todas las demás. Una estación sabe cuando una trama le va destinada porque lee todas las que le llegan y comprueba la dirección de destino. Tiene que rechazar todas las tramas con direcciones que no sean la suya. Sin embargo, también hay excepciones, y en algunos casos las estaciones también deben capturar tramas dirigidas a direcciones especiales (como las *multicast* y las *broadcast*).

Nota

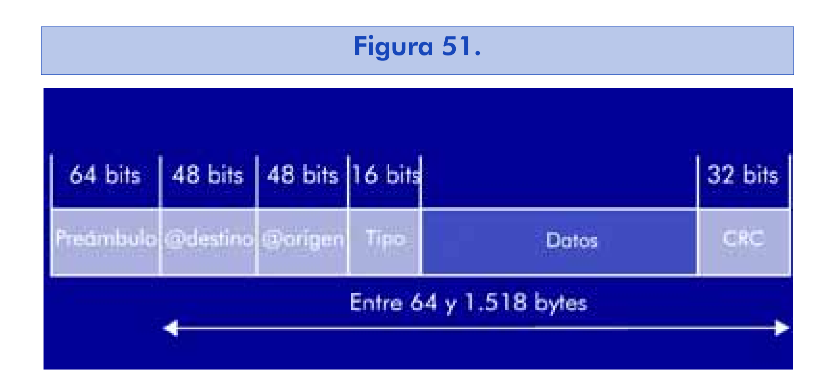
Uno de los casos en el que una estación no rechaza las tramas con direcciones diferentes de la suya es cuando la estación configura la tarjeta de red en modo promiscuo. En este modo, la tarjeta inhabilita su capacidad de filtrado y lee todas las tramas que pasan por la red. Equipos que, generalmente, funcionan en este modo son los puentes (*bridges*, sistemas destinados a la interconexión de LAN), los analizadores de tráfico (o analizadores de red) o los conmutadores (*switches*). No obstante, casi todas las tarjetas se pueden configurar en modo promiscuo, lo que con frecuencia es aprovechado por los ladrones de información (*hackers*) para leer y copiar información interesante que viaje por la LAN (principalmente contraseñas).

ANOTACIONES

11.3.1. Formato de la trama Ethernet

La manera de conocer las principales características de la trama Ethernet es ver los diferentes campos que la forman, que son los siguientes:

Figura 51.



a) **Preámbulo:** está formado por 64 bits, alternativamente 0 y 1. Los dos últimos son 11. Ello genera una señal cuadrada que permite a los terminales sincronizar adecuadamente los relojes de sincronismo de bit. Los dos últimos bits señalan dónde empieza la trama (sincronismo de trama). Su forma es idéntica en todas las tramas. Nosotros obviaremos su presencia en el resto de la explicación, puesto que sólo son una señal para marcar el inicio de la trama.

b) **Dirección de origen**: lleva la dirección física o dirección MAC del transmisor de la trama. Son 48 bits diferentes para cualquier terminal de la red Ethernet.

c) **Dirección de destino**: lleva la dirección MAC del destinatario especificada de la misma manera (en el mismo formato) que la dirección de origen. En este caso, sin embargo, tenemos tres tipos de direcciones posibles: *unicast*, *multicast* y *broadcast*.

d) **Tipo**: indica el tipo de contenido del campo de datos que lleva la trama (las tramas que transportan paquetes IP llevan un 0x800). Permite multiplexar diferentes protocolos dentro de una misma LAN. Xerox actualiza regularmente la lista de protocolos registrados (*Xerox Public Ethernet Packet Type*). Más adelante veremos las variedades de protocolos de Ethernet para conocer las variantes de Ethernet semicompatibles y saber cómo afecta su coexistencia a la manera como Ethernet ha tenido que definir el campo Tipo.

ANOTACIONES

Actividad

Consultad la lista de los protocolos registrados para haceros una idea de la cantidad de protocolos de red (los superiores a Ethernet) que hay.

e) **Datos**: se refiere al formato del campo de datos. Las restricciones sobre el tipo de datos que puede transportar Ethernet son las siguientes:

- La longitud de los datos debe ser múltiplo de 8 bits; es decir, Ethernet transporta la información en bytes. Ello no es ningún impedimento, puesto que los bits los envían sistemas que, por norma general, trabajan con bytes o múltiplos de bytes.

- Del mismo modo que PPP, Ethernet tiene limitada la longitud máxima de información transportable por la trama (MTU). En este caso, la MTU es de 1.500 bytes. Esta limitación tiene como objetivo evitar que una estación monopolice la LAN.

- El campo de datos debe tener como mínimo 46 bytes de longitud (en Gygabit Ethernet, 512). Ello se debe al hecho de que es preciso que la trama mínima Ethernet tenga 64 bytes (512 bits). Se considera que las tramas inferiores son resultado de colisiones y son obviadas por los receptores.

Este problema no se plantea en la variante de Ethernet IEEE802.3, puesto que este protocolo dispone de un campo de longitud y otro de relleno (*padding*) que permiten transmitir datos de hasta 1 byte de longitud, aunque la longitud que físicamente se transmite continúa siendo de 64 bytes.

Actividad

Imaginad qué sucedería si una estación quisiera enviar un fichero de 1 GB por la LAN dentro de una sola trama.

f) **CRC**: es un código de redundancia cíclica de 32 bits (CRC-32) para la detección de errores. Abarca toda la trama a excepción del preámbulo. Las tramas que no poseen un CRC correcto se ig-

ANOTACIONES

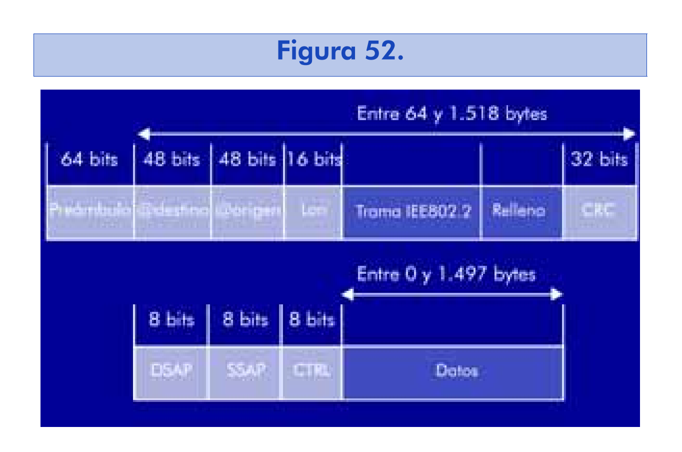
noran (como las tramas de menos de 64 bytes y las que no son múltiples de 8 bits).

Nota

Desde mediados de los años ochenta Ethernet ha convivido con una variante similar denominada *IEEE802.3* o *ISO8802.3*. Son estándares establecidos por organizaciones reconocidas (el IEEE y la ISO) dedicadas a la normalización (estándar *de iure*). Durante un tiempo se pensó que el IEEE802.3 acabaría sustituyendo a la Ethernet original (también denominada *Ethernet-DIX*, en honor a DEC, Intel y Xerox), que no podía transmitir tramas arbitrariamente pequeñas. Los protocolos que trabajan sobre Ethernet-DIX conocen esta limitación y llenan la trama hasta ocupar los 46 bytes de información.

El IEEE802.3 introduce un campo de longitud (en la misma posición en que Ethernet tiene el campo de tipo) que permite saber cuántos bytes útiles contiene el campo de datos. Si la longitud no llega a los 46 bytes mínimos, se llena con bytes (indefinidos) hasta llegar al mínimo. El receptor sólo debe leer el campo de longitud para extraer la información válida del mismo. El concepto de *tipo de Ethernet* (es decir, la coexistencia de diferentes protocolos por encima de Ethernet/IEEE802.3) se delega a un protocolo asociado: el IEEE802.2, protocolo de enlace que se puede utilizar en el IEEE802.3 y en otros protocolos de LAN y que posee unas funciones similares a las del HDLC.

Figura 52.



ANOTACIONES

Como en el nivel físico ambos estándares son totalmente compatibles, podríamos preguntarnos si pueden coexistir tramas Ethernet-DIX e IEEE802.3 dentro de una misma LAN. La respuesta es que sí y que no al mismo tiempo.

Fijaos en que todos los campos de la trama Ethernet y de la 802.3 son del mismo formato y significan lo mismo, a excepción del campo de tipo (Ethernet) y de longitud (802.3). Podríamos distinguirlos siempre que vigiláramos que ningún tipo Ethernet fuera equivalente a una longitud válida de 802.3. Los tipos con valores inferiores a 1.500 (0x5DC en hexadecimal) pueden confundirse con longitudes válidas.

Ello, obviamente, no podía tenerse en cuenta en la Ethernet-DIX original, puesto que es anterior al IEEE802.3. Por ello, apareció una adenda a la norma Ethernet-DIX, conocida como Ethernet-DIX-II, que elimina los identificadores de protocolos por debajo de 0x0600 (1.536 en decimal). Hoy día con frecuencia dentro de una misma LAN encontramos tramas Ethernet-DIX-II y tramas IEEE802.3.

El IP puede ir sobre cualquiera de los dos estándares, aunque casi nadie elige la posibilidad de encapsularlo sobre el IEEE802.3. El par de protocolos IEEE802.3 + 802.2 se utiliza en algunos de los sistemas operativos de red aparecidos en los años ochenta como, por ejemplo, el IPX de Novell y el NETBEUI de Microsoft.

ANOTACIONES

11.3.2. Tipos de medios físicos en Ethernet

Ethernet se ha ido adaptando a las necesidades del tiempo ampliando los subestándares de nivel físico. A continuación, mostramos una lista de los estándares más utilizados:

- 10Base2: alcance de 185 m, ó 925 m con repetidores, pero con coaxial delgado, flexible y barato (por ello, durante mu-

chos años esta red se ha denominado Cheapernet). aunque hoy día se tiende gradualmente a dejarlo de lado en favor de 10BaseT –que es mucho más fiable–, millones de terminales en todo el mundo están conectados con Ethernet-10Base2. Se utiliza sobre todo en topologías en bus.

- 10BaseT: conexión en estrella de las estaciones a un nudo central (concentrador o hub) por medio de par trenzado; la distancia máxima de la estación al concentrador es de 100 m. La distancia máxima entre estaciones se consigue encadenando cuatro concentradores, y es de 500 m.

  Representa una mejora importante respecto a los estándares anteriores, puesto que se centralizan en un solo punto la gestión y la monitorización del estado de toda la LAN. Asimismo, con las topologías en bus, el mal funcionamiento de una estación podía comportar el bloqueo de toda la red. Con 10BaseT, una mala conexión de un terminal es detectada por el concentrador, que simplemente la desconecta e indica que el enlace a la estación está cortado o inactivo (con una luz roja, por ejemplo).

- 10BaseF: similar a 10BaseT; sin embargo, en lugar de par trenzado, utiliza fibra óptica (generalmente, multimodo), con que se consigue un alcance mucho mayor (hasta 2 km).

- 100BaseT y 100BaseF: similares a 10BaseT y 10BaseF, respectivamente; sin embargo, funcionan a 100 Mbps. A causa del protocolo de control de acceso al medio CSMA/CD, el alcance se reduce mucho (100 m entre estación y concentrador, sin posibilidad de encadenar concentradores).

- Gigabit Ethernet: las variantes más comunes son 1000BaseT, sobre cableado de cobre categoría 5 (equivalente al necesario para 100BaseT) y 1000BaseSX, sobre fibra. Tiene el mismo alcance que 100BaseT, 100 m.

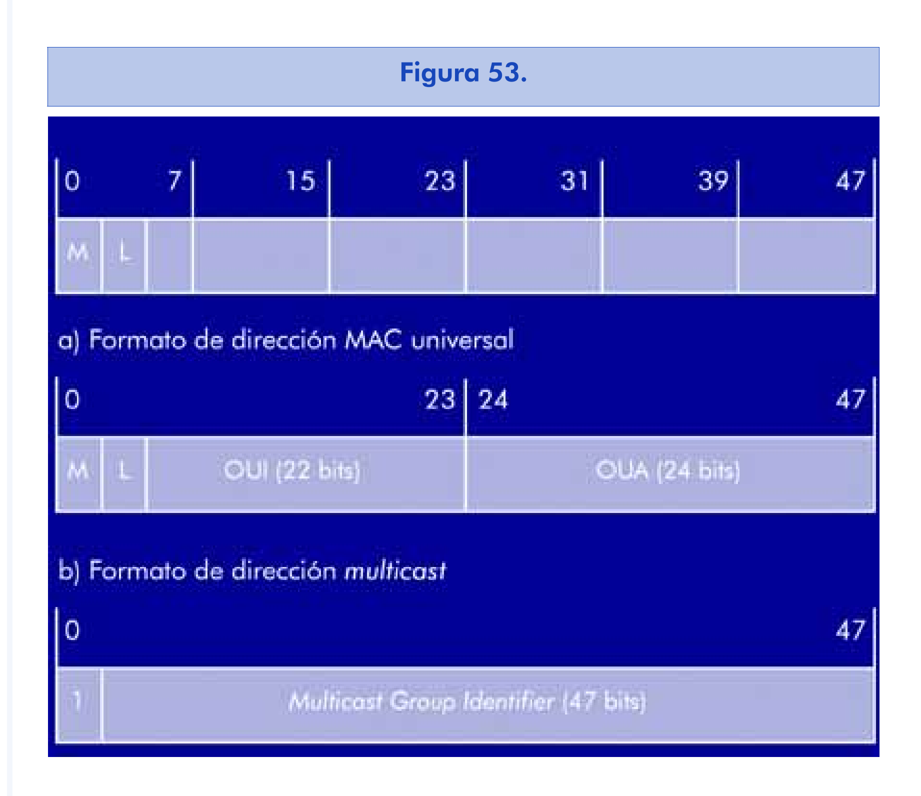
- 10 Gigabit Ethernet: actualización de Ethernet para el siglo XXI

ANOTACIONES

11.3.3. Direcciones LAN

Las direcciones LAN están divididas en diferentes campos, como puede observarse en la figura siguiente:

Figura 53.



La mitad menos significativa de la dirección (los bits del 2 al 23), asignada por el IEEE a cada fabricante de manera fija, es el OUI (*organizational unique identifier*). Este último, cuando fabrica las tarjetas, programa (en ROM) la dirección completa, que está formada por el OUI más una parte variable que el mismo fabricante asigna individualmente para cada tarjeta: la OUA (*organizational unique address*).

Existen dos bits del OUI que siempre son cero cuando se trata de la dirección de origen: el bit *multicast* (M) y el bit *local* (L). Este último no se utiliza casi nunca y, por tanto, lo consideraremos siempre cero.

Tanto la dirección de destino como la de origen de la trama tienen el mismo formato, con la única diferencia de que la dirección de destino también puede ser de tipo *multicast* (bit M = 1). En este caso, el número que lleva no se refiere a una estación en particular, sino que se dirige a un grupo de estaciones. Cada una de las cuales co-

Nota

No es del todo cierto que el único grupo *multicast* Ethernet relevante sea el *broadcast*. La red Internet dispone del protocolo IGMP (*Internet group multicast protocol*), que también trabaja sobre tramas Ethernet *multicast*.

ANOTACIONES

noce el grupo o grupos a los que está adscrita. Por norma general, cada uno de los grupos se refiere a grupos de estaciones que comparten una misma aplicación o un mismo protocolo. En el IP, el único grupo *multicast* Ethernet relevante es el *broadcast*.

Sobre papel, las direcciones LAN se escriben en hexadecimal, separando los bytes con dos puntos y escribiendo primero el byte menos significativo, por ejemplo:

08:00:00:10:97:00

El primer byte (que es el menos significativo) es siempre divisible por cuatro en direcciones no *multicast* que tienen los bits M y L a 0.

El grupo *broadcast* es especial, en el sentido de que, por defecto, todas las estaciones le pertenecen; por tanto, es una manera de difundir información simultáneamente a todas las estaciones. El concepto de *broadcast* no es exclusivo de Ethernet, sino que es común a muchos otros protocolos de LAN y WAN (también en el IP). Poner todos los bits de la dirección a 1 constituye la manera más habitual de representar la dirección *broadcast*, y es la que utilizan las LAN (FF:FF:FF:FF:FF:FF).

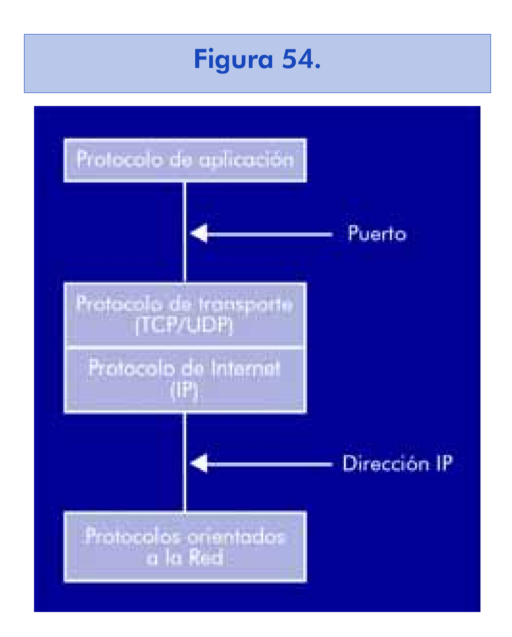
ANOTACIONES

12. Protocolos del nivel de transporte

El objetivo principal del nivel de transporte es establecer una comunicación extremo a extremo a través de una red. En otras palabras, actuar de interfaz entre los niveles orientados a la aplicación y los niveles orientados a la red de la jerarquía de protocolos (tanto OSI como TCP/IP).

El nivel de transporte oculta a los niveles altos del sistema el tipo de tecnología (red) al que está conectado el terminal. La figura siguiente describe el posicionamiento del nivel de transporte respecto al resto de los niveles:

Figura 54.



En este apartado nos interesan los dos protocolos del nivel de transporte que se definen en la pila TCP/IP: UDP y TCP. UDP es no orientado a la conexión, mientras que TCP es orientado a la conexión.

En el nivel de transporte se definen dos direcciones que lo relacionan con los niveles superior e inferior:

- La **dirección IP**, que ya conocemos, es la dirección que identifica un subsistema dentro de una red.
- El **puerto** identifica la aplicación que requiere la comunicación.

ANOTACIONES

Para identificar las diferentes aplicaciones, los protocolos TCP/IP marcan cada paquete (o unidad de información) con un identificador de 16 bits llamado *puerto*.

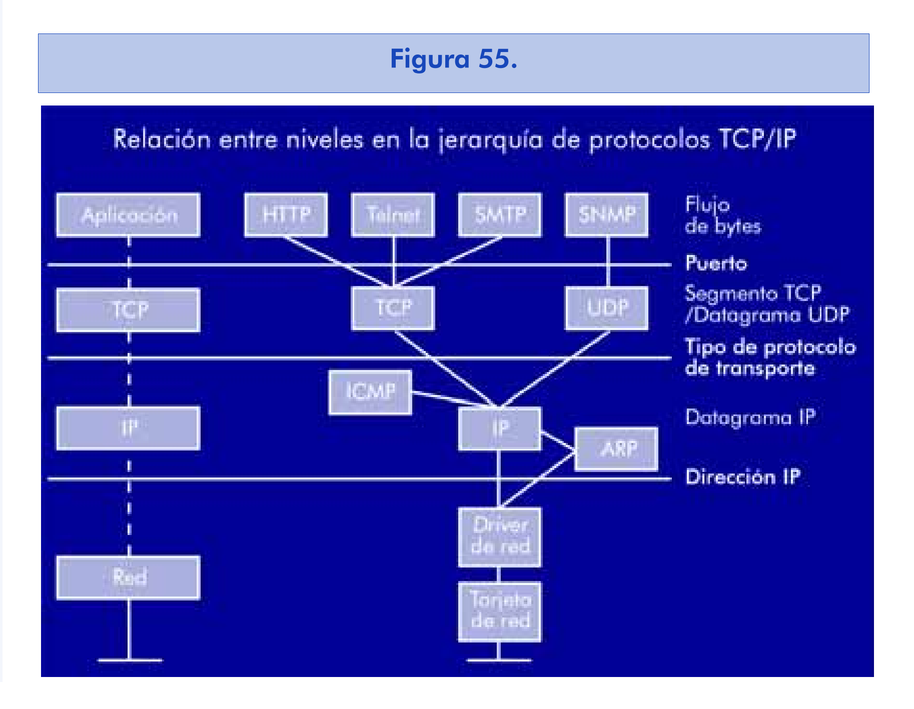
La verdadera utilidad de los puertos es que permiten multiplexar aplicaciones sobre protocolos del nivel de transporte. Ello significa que un mismo protocolo de transporte lleva información de diferentes aplicaciones y estas últimas son identificadas por el puerto.

Si alguna aplicación que corre en un terminal quiere establecer una comunicación con un servidor o con otro terminal, debe utilizar un protocolo de transporte: el TCP o el UDP. Como el destino puede encontrarse en una red remota, los protocolos de transporte necesitan el protocolo Internet para poder llegar al terminal o servidor remoto.



Cuando se establece la comunicación, no sólo es esencial conocer el puerto que identifica la aplicación de destino, sino también la dirección IP que identifica el terminal o servidor dentro del conjunto de redes.

Figura 55.



ANOTACIONES

Como podéis observar en la figura anterior, las aplicaciones utilizan uno de los dos protocolos de transporte para comunicarse con equipos remotos. Para que un protocolo de aplicación se pueda comunicar con otro del mismo nivel situado en un terminal remoto, debe transmitirle un flujo de bytes que es encapsulado por los protocolos del nivel de transporte.



El conjunto de bytes que transmite el nivel de transporte TCP se conoce como ***segmento TCP***, mientras que el conjunto de bytes que transmite el protocolo de transporte UDP se llama ***datagrama UDP***.

En general, dos aplicaciones se comunican siguiendo el modelo cliente/servidor. En una conexión es típico que una aplicación (el cliente) inicie una comunicación pidiendo una información a otra aplicación (el servidor). Pensemos en un ordenador que esté conectado a una LAN y tenga asignada una dirección IP. Supongamos que dicho ordenador actúa como servidor de correo electrónico, además de como servidor de nombres. Un cliente conectado a Internet que solicita resolver un nombre necesita conocer la dirección IP asignada a este ordenador y el puerto que identifica la aplicación servidor que resuelve nombres.

Nota

Veremos el modelo cliente/servidor en la unidad 15.

El cliente necesita conocer ambas direcciones puesto que el servidor estará conectado a una red y, por tanto, tendrá una dirección IP que debe ser conocida para que se pueda establecer una comunicación con esta máquina remota. Dicha comunicación se consigue por medio del IP. Sin embargo, una vez conseguida, el servidor debe ser capaz de identificar la aplicación con que desea comunicarse el cliente entre las muchas que corren: servidor de nombres, servidor de correo electrónico, etc.

El cliente conoce la dirección IP de origen (la suya), la dirección IP de destino (la del servidor) y su puerto de origen (identifica la aplicación cliente). Sin embargo, también debe conocer el número (puerto de destino) que identifica la aplicación deseada en el servidor, y lo hace por medio de los llamados *puertos conocidos* (*well-known port*).

ANOTACIONES



Un **puerto conocido** (*well-known port*) es un puerto (número) reservado que identifica una aplicación conocida. Estos puertos son asignados por IANA (Internet Assigned Numbers Authority)

Ejemplo

Los valores de puertos conocidos para aplicaciones que utilizan el UDP son los siguientes:

- Puerto 7 para el servidor de eco.
- Puerto 53 para el servidor de nombres (DNS, *domain name server*).
- Puerto 69 para el protocolo de transferencia de ficheros trivial (TFTP, *trivial file transfer protocol*).

Y algunos valores de puertos conocidos para aplicaciones que utilizan el TCP son los siguientes:

- Puertos 20 y 21 para el protocolo de transferencia de ficheros, FTP de datos y de control respectivamente.
- Puerto 23 para el *Telnet Remote Login*.
- Puerto 80 para el HTTP.

Evidentemente, el servidor no necesita conocer *a priori* el puerto de origen, puesto que se limita a responder a cualquier petición de cualquier cliente. Este último informa en la unidad de datos de protocolo (PDU) del nivel de transporte (o bien un datagrama UDP, o bien un segmento TCP) de los puertos de origen y de destino, de manera que el servidor conocerá el puerto de origen una vez haya recibido una petición.

ANOTACIONES

13. El UDP (*user datagram protocol*)

El UDP es un protocolo no orientado a la conexión, de manera que no proporciona ningún tipo de control de errores ni de flujo, aunque utiliza mecanismos de detección de errores. En caso de detectar un error, el UDP no entrega el datagrama a la aplicación, sino que lo descarta. Conviene recordar que, por debajo, el UDP está utilizando el IP, que también es un protocolo no orientado a la conexión. Las características más importantes del UDP son las siguientes:

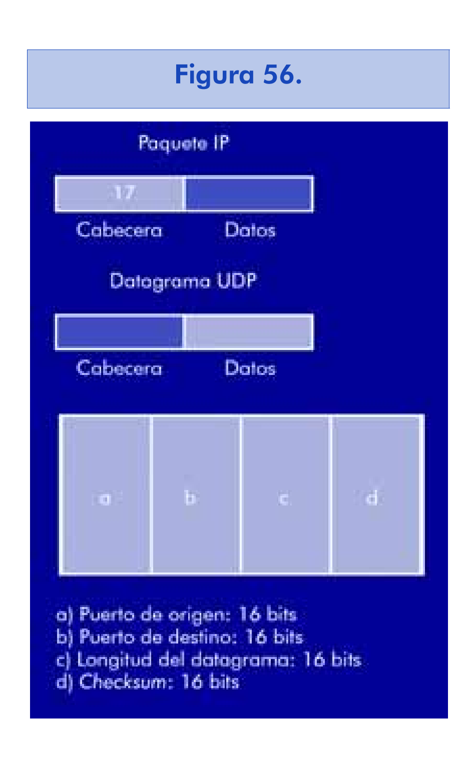
Nota

El UDP es un protocolo no orientado a la conexión. Ello significa que cada datagrama UDP existe con independencia del resto de los datagramas UDP.

- No garantiza la fiabilidad; es decir, no ofrece la seguridad de que cada datagrama UDP transmitido llegue a su destino; es un protocolo *best-effort*: el UDP hace todo lo posible para transferir los datagramas de su aplicación, pero no garantiza su entrega.

- No preserva la secuencia de la información que le proporciona la aplicación. Como está en modo datagrama y utiliza un protocolo por debajo como el IP, que también está en modo datagrama, la aplicación puede recibir la información desordenada. La aplicación debe estar preparada para que haya datagramas que se pierdan, lleguen con retardo o se hayan desordenado.

Figura 56.



ANOTACIONES

La figura anterior muestra la unidad de datos del protocolo UDP y su encapsulamiento en un paquete IP. Cada operación de salida de un datagrama UDP provoca la generación de un paquete IP.

El datagrama UDP consta de una cabecera y un cuerpo para encapsular los datos. La cabecera consta de los elementos siguientes:

Nota

Hay muchas aplicaciones que limitan la medida de sus buffers de transmisión y recepción por debajo de la medida máxima de un datagrama UDP. Por ejemplo, es típico encontrar aplicaciones que proporcionan, por defecto, medidas máximas del datagrama UDP de 8.192 bytes. Este valor proviene de la cantidad de datos del usuario que el NFS (*network file system*) puede leer o escribir por defecto.

- Los **campos Puerto de origen y Puerto de destino**, que identifican las aplicaciones en los terminales de origen y de destino. Cada puerto tiene 16 bits.

- El **campo Longitud** indica la longitud, en bytes, del datagrama UDP incluyendo la cabecera UDP (es la diferencia de la longitud del datagrama IP menos la cabecera IP). Como la longitud máxima de un datagrama IP es de 65.535 bytes, con una cabecera estándar de 20 bytes, la longitud máxima de un datagrama UDP es de 65.515 bytes.

- El **campo *Checksum*** (16 bits) es opcional y protege tanto la cabecera como los datos UDP (es preciso recordar que el *checksum* del datagrama IP sólo cubre la cabecera IP). Cuando el UDP recibe un datagrama y determina que hay errores, lo descarta y no lo entrega a ninguna aplicación.

Nota

El cálculo del *checksum* en el UDP es muy parecido al cálculo del *checksum* en el IP (suma en complemento a 1 de palabras de 16 bits), con la particularidad de que la longitud del datagrama UDP puede ser par o impar. En caso de que sea impar, se le añade un 0 al final del datagrama para calcular el *checksum* (este 0 no se transmite). Para calcular el *checksum*, el UDP utiliza una seudocabecera de 12 bytes con algunos de los campos IP. Esta última no se transmite; el UDP sólo la utiliza para calcular el *checksum* y le sirve para comprobar que la información que le proporciona el IP sea realmente para él.

ANOTACIONES



Si el UDP no aporta nada a IP, se podría pensar que su presencia es superflua. Pero no es así, porque no es estrictamente cierto que no aporte nada. Aporta la multiplexación de aplicaciones sobre la misma comunicación de red, a través del concepto de puerto.

Las aplicaciones que no requeiren de la funcionalidad del TCP, usan el UDP como protocolo de trasporte. Podemos poner dos ejemplos de estas aplicaciones:

- Aplicaciones en tiempo real. Estas aplicaciones requieren poco retardo (mejor dicho, poca variabilidad en el retardo), y TCP puede introducir retardos considerables si tiene que esperar, por ejemplo que le llegue un paquete que se ha perdido.

- Aplicaciones interesadas en transmitir información en modo *multicast* o *broadcast* (a un grupo de usuarios o a todos los de una red). En este caso, no tiene sentido establecer una conexión como hace el TCP con cada una de las estaciones destino.

14. El TCP (*transmission control protocol*)

Como hemos podido observar, el UDP no garantiza la entrega de la información que le proporciona una aplicación. Tampoco reordena la información en caso de que llegue en un orden diferente de aquél en que se ha transmitido. Existen aplicaciones que no pueden tolerar dichas limitaciones. Para superarlas, el nivel de transporte proporciona un protocolo llamado *TCP*.

El TCP proporciona fiabilidad a la aplicación; es decir, garantiza la entrega de toda la información en el mismo orden en que ha sido transmitida por la aplicación de origen. Para conseguir esta fiabilidad, el TCP proporciona un servicio orientado a la conexión con un control de flujo y errores.

14.1. El TCP proporciona fiabilidad

Para proporcionar un servicio fiable a la aplicación, el TCP se basa en los principios siguientes:

1) **Transmisión libre de error**. El TCP debe entregar a la aplicación de destino exactamente la misma información que le entregó la aplicación de origen. De hecho, se trata de una entrega "casi libre" de errores, puesto que puede haber algunos que un mecanismo de detección de errores no pueda detectar.

2) **Garantía de entrega de la información**. El TCP garantiza que toda la información transmitida por la aplicación de origen se entregue a la aplicación de destino. Si no es posible, el TCP debe avisar a la aplicación.

ANOTACIONES

3) **Garantía de mantenimiento de la secuencia de transmisión.** El TCP garantiza la entrega del flujo de información en el mismo orden en que le fue entregado por la aplicación de origen.

4) **Eliminación de duplicados.** El TCP garantiza que sólo entregará una copia de la información transmitida a la aplicación de destino. En caso de que reciba copias a causa del funcionamiento de la red o de los protocolos que se implementan por debajo del nivel de transporte, el TCP las eliminará.

La fiabilidad de la transmisión que proporciona TCP se consigue gracias a las siguientes estrategias:

- El TCP está orientado a la conexión: tiene una fase de establecimiento de la conexión, una de transmisión de datos y una de desconexión.

- El TCP utiliza el concepto *buffered transfer*: cuando se transfiere información, el TCP divide los flujos de datos (*byte stream*) que le pasa la aplicación en segmentos del tamaño que le convenga. El TCP decide el tamaño de los segmentos tanto si la aplicación genera un byte de información, como si genera flujos de grandes dimensiones. En el primer caso, el TCP puede esperar a que la memoria intermedia se llene más antes de transferir la información, o bien la puede transferir de inmediato (mecanismo *push*). En caso de que los flujos sean muy grandes, el TCP puede dividir la información en tamaños más pequeños antes de transferirlos.

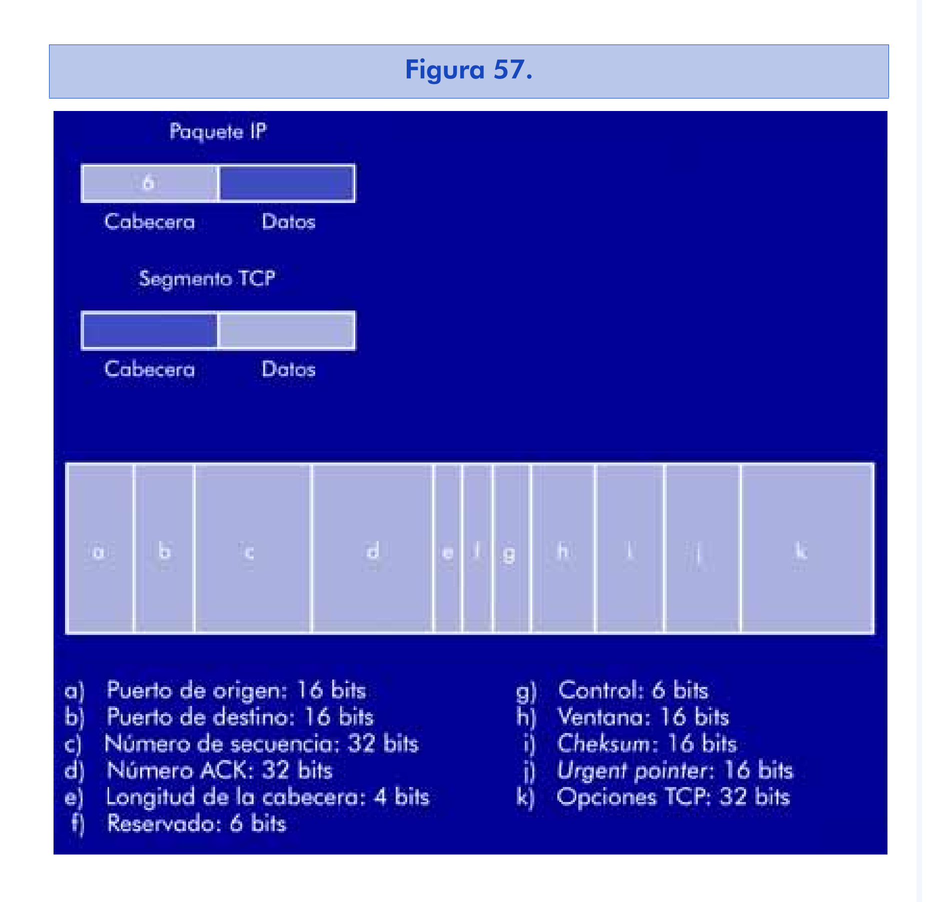
- El TCP utiliza una conexión *full duplex*: la transferencia de información es en ambos sentidos. La aplicación ve dos flujos independientes. En caso de que la aplicación cierre uno de los flujos, la conexión pasa a ser *half duplex*. Ello significa que uno de los extremos (el que no ha cerrado la conexión) puede continuar enviando información por el canal, mientras que el otro extremo (el que ha cerrado la conexión) se limita a reconocer la información. No obstante, no es normal encontrar este caso. Lo más habitual es que, si un extremo cierra la conexión, el otro también la cierre.

ANOTACIONES

14.2. Formato del segmento TCP

La unidad de información del protocolo TCP se llama *segmento TCP* y su formato el siguiente:

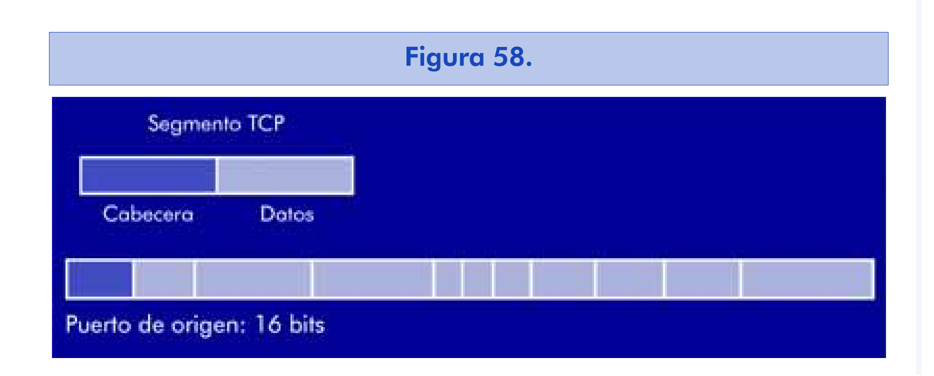
Figura 57.



El segmento TCP consta de una cabecera y un cuerpo para encapsular datos. La cabecera consta de los campos siguientes:

a) El campo **Puerto de origen** identifica la aplicación en el terminal de origen.

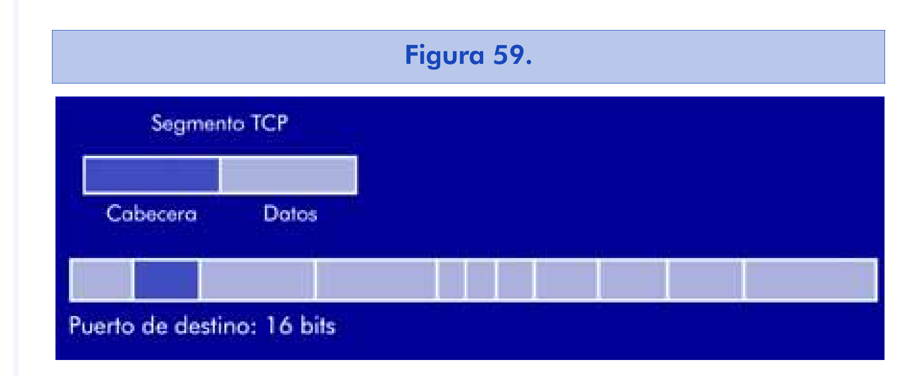
Figura 58.



ANOTACIONES

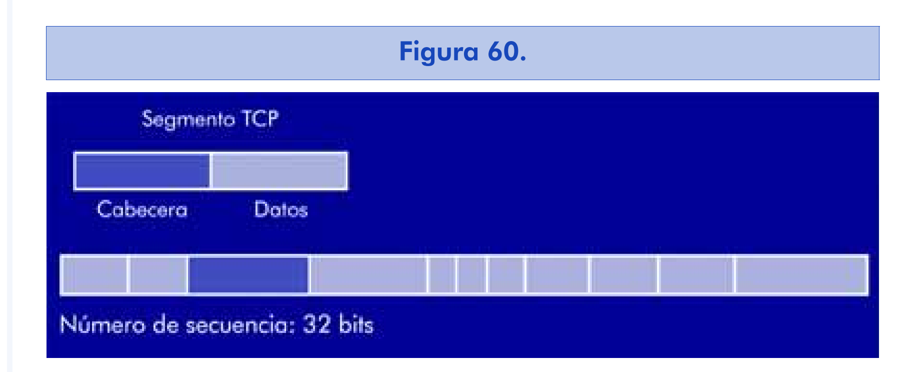
b) El campo **Puerto de destino** identifica la aplicación en el terminal de destino.

Figura 59.



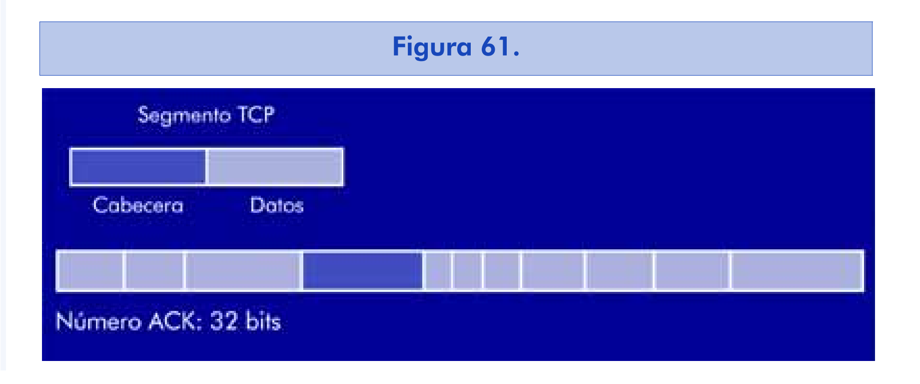
c) El campo **Número de secuencia** identifica el primer byte del campo de datos. En el TCP no se numeran segmentos, sino bytes. Por tanto, el número de secuencia identifica el primer byte de los datos que envía el segmento: al principio de la conexión se asigna un número de secuencia inicial (ISN, del inglés *initial sequence number*), a partir del cual el TCP numera los bytes consecutivamente.

Figura 60.



d) El campo **Número ACK**. El TCP reconoce datos por medio de la técnica de *piggybacking*. Al activar un bit de la cabecera (el bit ACK), el TCP tiene en cuenta el número de secuencia ACK que indica al otro extremo TCP el próximo byte que está dispuesto a recibir. Dicho de otra manera, el número ACK menos uno indica el último byte reconocido.

Figura 61.



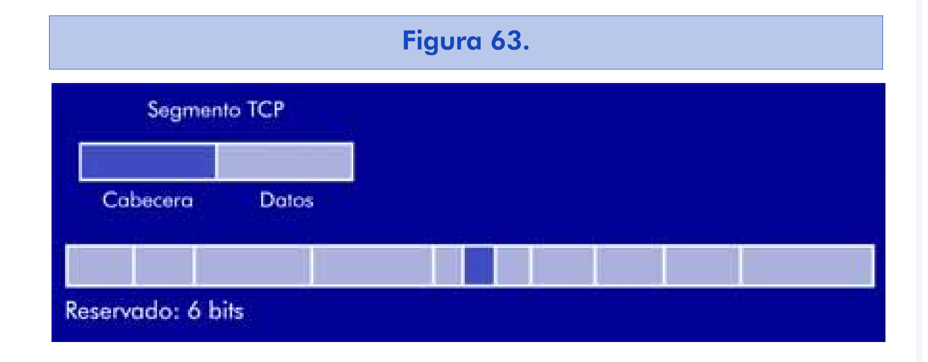
ANOTACIONES

e) El campo **Longitud de la cabecera** indica la longitud de la cabecera, que puede ser variable. La longitud típica es de 20 bytes; sin embargo, si el TCP utiliza el campo de opciones, puede llegar a una longitud máxima de 60 bytes. De este modo, el TCP sabe dónde empiezan los datos.

Figura 62.

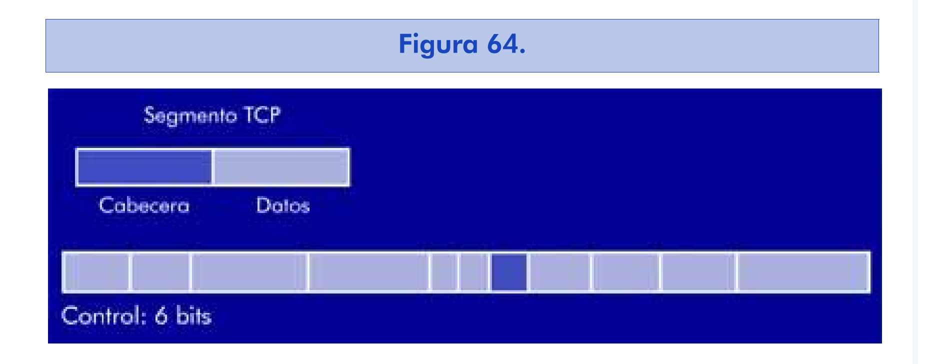
f) El campo **Reservado**, tal como su nombre indica, está reservado y se inicializa con ceros.

Figura 63.



g) El campo **Control** está formado por seis indicadores independientes, cada uno de los cuales señala una función específica del protocolo cuando está activo (a 1):

Figura 64.



- **URG**: indica que hay datos urgentes (y el campo *Urgent pointer* indica la cantidad de datos urgentes existentes en el segmento).

Nota

No debe confundirse PSH con el indicador URG, que indica que la aplicación ha señalado una porción del segmento como urgente.

- **ACK**: cuando este bit está activo, el campo Número ACK indica el byte siguiente que espera recibir la conexión TCP. Si este bit no está activo, el campo Número ACK no tiene ningún significado para el TCP.

- **PSH**: invoca la función *push* en el protocolo. Esta función dice al receptor que entregue a la aplicación todos los datos que tenga disponibles en la memoria intermedia de recepción sin esperar a completarlos con datos adicionales. De este modo, los datos no esperan en la memoria intermedia receptora hasta completar un segmento de dimensión máxima.

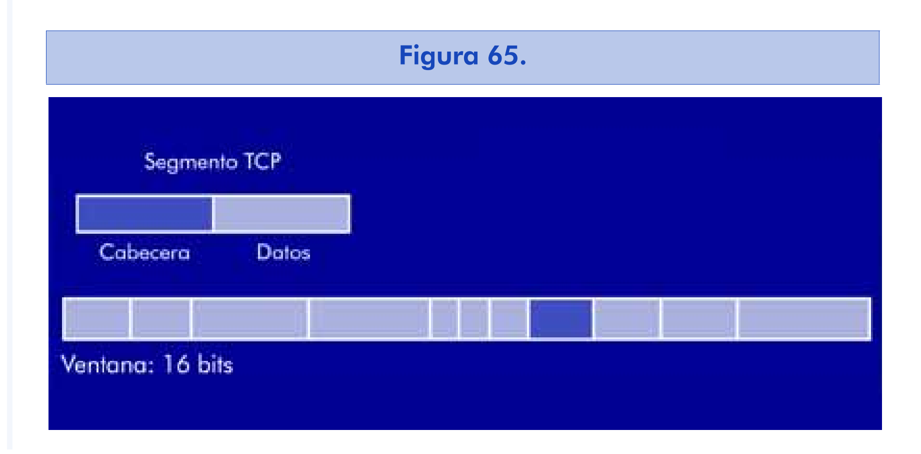
- **RST**: realiza un *reset* de la conexión.

- **SYN**: se utiliza para iniciar una conexión y también sirve para resincronizar los números de secuencia.

- **FIN**: indica que el transmisor ha acabado la conexión.

h) El campo **Ventana** indica cuántos bytes componen la ventana de transmisión del protocolo de control de flujo por ventana deslizante. A diferencia de los protocolos del nivel de enlace, en que la ventana era constante y contaba tramas, en el TCP la ventana es variable y cuenta bytes. Con cada segmento transmitido, un extremo TCP advierte al otro extremo de la cantidad de datos que está dispuesto a recibir en cada momento. De este modo, el extremo que recibe un segmento actualiza el tamaño de su ventana de transmisión.

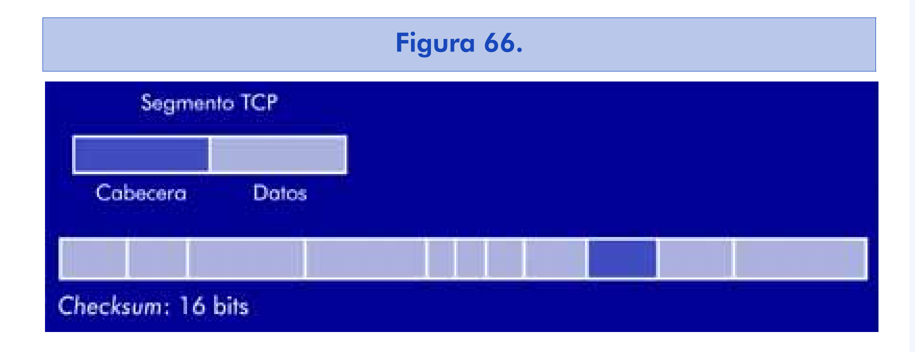
Figura 65.



ANOTACIONES

i) El campo ***Checksum*** se utiliza para detectar errores.

Figura 66.

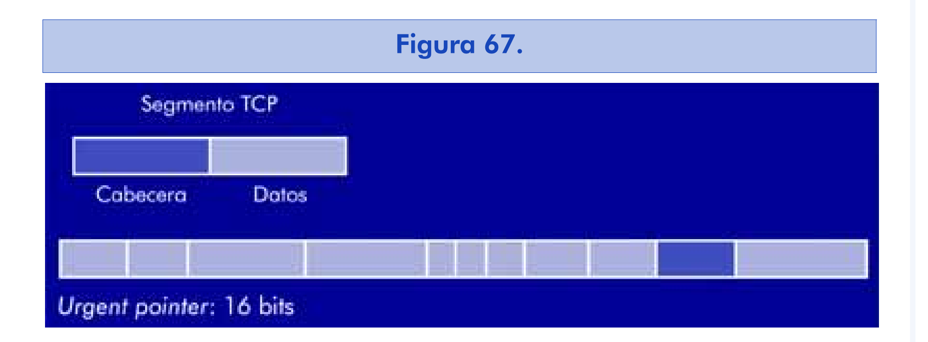


j) El campo *Urgent pointer* tiene sentido cuando el bit de control URG está activo. Indica que los datos que envía el origen son urgentes e identifica el último byte de dicho campo. La aplicación es la que indica que estos últimos son urgentes y lo sabe porque el TCP se lo indica en la recepción.

Ejemplo

Si el número de secuencia indica 1.000 y el urgent pointer indica 200, significa que los bytes del 1.000 al 1.200 se consideran urgentes. A partir del byte 1.201 los datos se vuelven a considerar normales.

Figura 67.

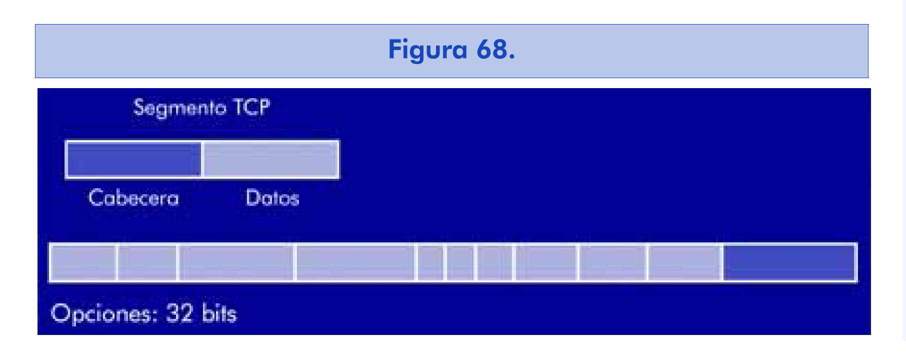


Nota

Algunas aplicaciones que utilizan el *Urgent pointer* son, por ejemplo `telnet`, `rlogin` o `ftp`. En la librería de sockets, el tráfico urgente se denomina tráfico fuera de banda (*out of band*).

k) El campo **Opciones TCP** permite añadir campos a la cabecera para realizar las operaciones siguientes:

Figura 68.



ANOTACIONES

Nota

El tamaño máximo del segmento TCP transmitido se especifica durante el establecimiento de la conexión y define la máxima longitud de datos que enviará el TCP.

- Marcar el tiempo (*timestamp*) en que se transmitió el segmento y de este modo poder monitorizar los retardos que experimentan los segmentos desde el origen hasta el destino.
- Aumentar el tamaño de la ventana.
- Indicar el tamaño máximo del segmento (MSS, del inglés *maximum segment size*) que el origen está preparado para recibir. Por tanto, el receptor no le puede transmitir segmentos por encima de este valor.

Actividad

¿Cuál es el tamaño de un datagrama IP en función de MSS?

Solución

Si el tamaño de los datos TCP es MSS, será preciso añadirle 20 bytes de la cabecera TCP más 20 bytes de la cabecera IP (teniendo en cuenta las cabeceras básicas sin opciones). Ello significa que la longitud del datagrama IP será de MSS + 40 bytes (siempre asumiendo que tanto el TCP como el IP no utilizan sus campos de opciones).

Si no se especifica el tamaño máximo durante la transmisión del segmento SYN, se toman por defecto 536 bytes (el tamaño por defecto de un datagrama IP es de 576 bytes, menos los 40 bytes de las cabeceras IP y TCP).

Nota

Consultad la MTU en el apartado 11.1.2.

El hecho de elegir el MSS no es trivial. En general, cuanto mayor sea el MSS, mejor, puesto que las cabeceras IP y TCP se amortizan más. Sin embargo, si la MTU es pequeña, será preciso fragmentar el datagrama IP (es decir, el segmento TCP); por tanto, por norma general no interesa elegir MSS mayores que la MTU. En este caso, existen diferentes posibilidades:

1) Buscar la MTU local de la red a que está conectada la estación y, si hay MTU más pequeñas hasta el destino, habrá fragmentación.

2) Utilizar *MTU discocvery path*, un mecanismo de búsqueda para averiguar cuál es la MTU menor desde el origen hasta el destino y utilizar como MSS esta última menos los 40 bytes de cabeceras IP y TCP.

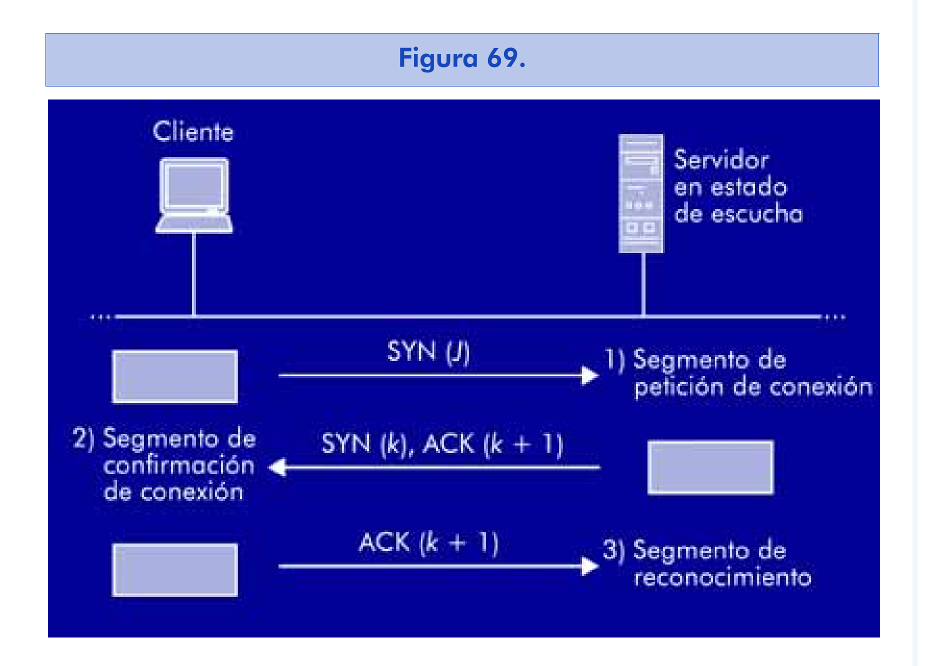
ANOTACIONES

14.3. Establecimiento de la conexión

Para establecer una conexión, el TCP utiliza el **protocolo *three-way handshake***. Este último necesita tres segmentos TCP para poder establecer la conexión.

Consideremos que el servidor está en un estado de escucha, llamado *listen*, y que el cliente quiere establecer una conexión con el servidor. El TCP de la máquina cliente iniciará la petición de conexión TCP, que será contestada por el TCP de la máquina servidor.

Figura 69.



Para que el cliente TCP pueda establecer una conexión TCP con el servidor, se siguen los pasos siguientes:

1) Petición de la conexión

El TCP cliente envía un segmento de petición de conexión al servidor. Dicho segmento, que se conoce como *segmento SYN* porque tiene activado el bit SYN en el campo Control de la cabecera del segmento TCP, especifica el número de secuencia inicial TCP del cliente (ISN).

El número de secuencia inicial se elige al azar. La razón es muy sencilla. Hay paquetes que pueden sobrevivir en la red una vez se ha cerrado la conexión TCP (incluso si ha sido a causa de una caída del

Nota

El segmento SYN especifica más parámetros, tales como el puerto del servidor al que se quiere conectar el cliente, y suele especificar también la medida máxima del segmento (MSS) que el cliente transmitirá.

sistema). Es preciso asegurarse de que una conexión nueva elige un número de secuencia inicial que no exista. El TCP recomienda utilizar un número de secuencia inicial basado en una variable que se incrementa una cantidad *x* cada *y* tiempo (por ejemplo, en 4.4BSD hay un contador que se incrementa cada 8 ms).

Si el sistema cae, pasados unos segundos vuelve a estar en funcionamiento e inmediatamente se establece una conexión nueva utilizando el mismo puerto y la misma dirección IP, se podría interpretar que los segmentos TCP que han quedado retrasados en la red y que ya existían con anterioridad a la caída de la máquina, pertenecen a la conexión nueva, lo que provocaría la confusión y el mal funcionamiento de dicha conexión. Ello sucedería incluso con independencia del número de secuencia inicial elegido.

Con el objetivo de protegerse de esta situación, se combinan dos técnicas: una consiste en elegir el número de secuencia inicial de manera aleatoria y la otra es el denominado *quiet time*, que consiste en que el TCP no crea ninguna conexión nueva después de un rebote de máquinas hasta que no transcurre un tiempo determinado denominado **MSL** (del inglés *maximum segment lifetime*: "tiempo máximo de vida de un segmento"). De este modo, se asegura de que no recibirá segmentos antiguos de otras conexiones.

Nota

El MSL depende de la implementación; sin embargo, los valores normales son, aproximadamente, de 30 segundos, 1 minuto ó 2 minutos. No obstante, existen muchas implementaciones que no tienen en cuenta esta situación, puesto que consideran que un rebote de máquinas dura más tiempo que el MSL.

2) Confirmación de la conexión

El servidor responde a la petición de establecimiento de la conexión con un segmento SYN que indica el número de secuencia inicial que utilizará.

Asimismo, este segmento contiene un reconocimiento (ACK) del segmento SYN del cliente que indica el ISN del cliente más 1 (el número de secuencia inicial del cliente más 1).

Nota

Conviene recordar que el TCP numera los ACK con el número de secuencia del próximo byte que espera recibir (en este caso, el servidor espera que el próximo byte enviado por el cliente sea J + 1). En la figura anterior, sería el segmento SYN (K), ACK (J + 1), donde K es el ISN elegido por el TCP servidor.

3) Reconocimiento de la conexión

El cliente reconoce el segmento SYN (*K*) del servidor con un reconocimiento que contiene el ISN servidor más 1. En la figura anterior, sería el segmento ACK (*K* + 1).

Se dice que quien envía el primer segmento SYN (en este caso, el cliente) efectúa una apertura activa (*active open*), mientras que quien recibe el primer segmento SYN y envía el próximo segmento SYN (en este caso, el servidor) lleva a cabo una apertura pasiva (*passive open*).

Puede darse el caso de que ambos extremos efectúen una apertura activa en el mismo momento. Esta situación se denomina *apertura simultánea* (*simultaneous open*).

Después de estos tres pasos, podemos decir que ya se ha establecido la conexión entre el cliente y el servidor.

Nota

Utilizaremos el programa `tcpdump` para ver cómo funciona el protocolo de establecimiento de una conexión.

Asumimos que nos hemos conectado a una máquina llamada *argos* y hacemos un `rlogin` a la máquina *helios*

```
argos % rlogin helios
```

Nota

En el anexo 2 podéis encontrar una descripción del programa `tcpdump`.

ANOTACIONES

Las primeras líneas que obtenemos con el tcpdump son las siguientes:

• 15:56:54.796091 argos.1023 > helios.login: S 3541904332: 3541904332 (0) win 31744 <mss 1460>

• 15:56:54.796091 helios.login > argos.1023: S 548133143: 548133143 (0) ack 33541904333 win 8760 <mss 1460>

• 15:56:54.796091 argos.1023 > helios.login: . ack 548133144 win 31744

Interpretación

1) *argos*, desde el puerto 1.023, envía a helios una petición de conexión por medio de un segmento SYN. El número de secuencia inicial (ISN) elegido por *argos* es el 3.541.904.332, y *argos* anuncia que puede recibir 31.744 bytes sin reconocerlos y que espera recibir segmentos con un tamaño máximo de 1.460 bytes.

2) *helios* responde con un segmento SYN eligiendo como ISN el número 548.133.143 y responde con un ACK con el byte siguiente que espera recibir de *argos*, el 3.541.904.333 (3.541.904.332 + 1). Anuncia que puede recibir 8.760 bytes y que espera recibir segmentos con un tamaño máximo de 1.460 bytes.

3) *argos* reconoce el segmento SYN con un segmento en el que espera recibir el byte 548.133.144 (548.133.143 + 1), *argos* vuelve a advertir que está dispuesto a recibir hasta 31.744 bytes.

A continuación, empezaría el intercambio de información entre el cliente y el servidor (por ejemplo peticiones de *login,* contraseña, *prompt* de la máquina, etc.).

Actividad

Utilizad el programa tcpdump para ver el establecimiento de una conexión. Con esta finalidad, estableced una conexión con aplicaciones diferentes (telnet, ftp, rlogin, etc.) y monitorizad la conexión. Observad los segmentos de inicio de la conexión, el valor del número de secuencia inicial, el del número ACK inicial y el tamaño de la ventana.

ANOTACIONES

14.4. Terminación de la conexión

Cuando la transferencia de la información ha finalizado, el TCP dispone de un protocolo de terminación de la conexión para cerrarla.

En una conexión TCP *full duplex*, en la que los datos fluyen en ambos sentidos, independientes el uno del otro, cualquier conexión debe cerrarse independientemente.

Es preciso tener en cuenta que tanto el cliente como el servidor pueden cerrar la conexión. Sin embargo, la situación normal es que la aplicación cliente inicie la petición de conexión y tenga, posiblemente, un usuario interactivo que le pida su cierre por medio, por ejemplo, de una instrucción, que en telnet sería `logout` y en un ftp sería `bye`. Por tanto, supongamos que es el cliente quien pide cerrar la conexión (si fuera el servidor, sería similar). Los pasos que se siguen son los siguientes:

1) El cliente envía un segmento TCP del tipo FIN con el número de secuencia correspondiente (*J*). Ello significa que a partir de este momento no habrá más datos que fluyan en este sentido (cliente → servidor).

2) El servidor envía una confirmación del cierre por medio de un ACK con el número de secuencia recibido más 1 ($J + 1$).

El TCP servidor indica a su aplicación que el cliente cierra la conexión. La aplicación servidor indica a su TCP que la cierre a continuación.

3) El servidor envía un segmento TCP del tipo FIN al cliente con el número de secuencia correspondiente (*K*).

4) El TCP cliente responde automáticamente con un ACK ($K + 1$).



Se dice que quien envía el primer segmento FIN (en este caso el cliente) lleva a cabo un cierre activo (*active close*), mientras que quien lo recibe (en este caso el servidor) realiza un cierre pasivo (*passive close*).

Nota

El segmento FIN recibe este nombre porque tiene activado el bit FIN en el campo Control de la cabecera del segmento TCP.

ANOTACIONES

La conexión que efectúa el cierre activo entra en un estado denominado *TIME_WAIT*, de manera que deberá esperar un tiempo (por norma general, una o dos veces el MSL) antes de utilizar de nuevo el mismo puerto. Lo más habitual es que sea el cliente quien efectúe el cierre activo. Como los clientes suelen utilizar puertos locales efímeros, este tiempo de espera no les afecta. En cambio, si es el servidor quien efectúa el cierre activo, podemos encontrarnos con que no se pueda reinicializar durante 1 ó 2 minutos. Ello sucede porque el servidor utiliza puertos conocidos que no se pueden volver a reasignar hasta que no acabe el procedimiento *quiet time* y se salga del estado *TIME_WAIT*.

Lectura complementaria

Podéis encontrar una sección dedicada a este tema en el libro:

W.R. Stevens (1998). *TCP/IP Illustrated* (vol. 1: "The Protocols", cap. 19, pág. 252). Wilmington: Addison-Wesley, 1994.

Es posible que sólo cierre la conexión (salida de datos) uno de los extremos, mientras que el otro (recepción) se mantiene abierto. Esta situación se denomina *half-close*, pero hay pocas aplicaciones que se aprovechen de ella. Lo más normal es que ambas aplicaciones cierren sus canales de comunicaciones. Asimismo, puede darse el caso de que dos extremos efectúen un cierre activo. Esta situación se denomina *cierre simultáneo* (*simultaneous close*).

Monitorización de la terminación de una conexión utilizando el programa `tcpdump`

Utilizaremos el comando `tcpdump` para ver cómo funciona el protocolo de terminación de la conexión. Asumimos que en el rlogin del ejemplo de establecimiento de la conexión *helios* hace un `logout` (pide el cierre de la conexión).

```
helios % logout
```

Las líneas que obtenemos con el `tcpdump` son las siguientes:

```
15:57:01.616091 helios.login > argos.1023: F 1417: 1417 (0)
ack 41 win 8760

15:57:01.616091 argos.1023 > helios.login: .ack 1418 win
31744

15:57:01.616091 argos.1023 > helis.login: F 41:41 (0) ack
580 31744

15:57:01.616091 helios.login > argos.1023: .ack 42 win 8760
```

ANOTACIONES

Interpretación

1) *helios* envía un segmento con el indicador F (FIN). El número de secuencia es el 1.417 y envía 0 bytes de datos. Espera recibir el byte 41 de *argos* y advierte una ventana de 8.760 bytes.
2) *argos* envía un reconocimiento por medio de un segmento con ACK 1.418 (1.417 + 1) y advierte una ventana de 31.744 bytes.
3) Ahora le toca a *argos* cerrar su conexión TCP. Con esta finalidad, envía un segmento con el indicador F (FIN) a *helios*. El número de secuencia es el 41 (es el que esperará *helios*) y envía 0 bytes de datos. Advierte una ventana de 31.744 bytes.
4) *helios* recibe el segmento, lo reconoce con el ACK numerado como 42 (41 + 1) y advierte una ventana de 8.760 bytes.

helios ha efectuado un cierre activo, mientras que *argos* ha efectuado un cierre pasivo.

Nota

La notación de los números de secuencia y números ACK se establece a partir del ISN; es decir un número de secuencia 1.417 indica un número de secuencia ISN+1.417.

Actividad

Utilizad el programa `tcpdump` para ver el cierre de una conexión. Con esta finalidad, estableced una conexión con diferentes aplicaciones (`telnet`, `rlogin`, etc.) y supervisadla.

14.5. Diagrama de estados del TCP

En el diagrama de estados del TCP se describen los diferentes estados por los que pasa una conexión desde su establecimiento hasta su terminación, incluyendo la etapa de transferencia de la información. Los nombres de los estados TCP son los mismos que se pueden consultar con la llamada al sistema *netstat*.

Lectura complementaria

W.R. Stevens (1998). "The Protocols" *TCP/IP Illustrated* (vol. 1). Wilmington: Addison-Wesley, 1994.

El estado ESTABLISHED se corresponde con la transferencia de la información. El resto de los estados se corresponden con el establecimiento y la terminación de la conexión, teniendo en cuenta todas las maneras posibles de establecer y cerrar una conexión en el TCP. Los

ANOTACIONES

símbolos SYN, RST, FIN y ACK se corresponden con los bits de indicación de la cabecera TCP.

Figura 70.

Un ejemplo de cómo se interpreta este diagrama sería el protocolo de terminación de una conexión, en que ya vimos que el extremo TCP que pedía el cierre efectuaba un cierre activo. Ello significa que pasaría del estado ESTABLISHED al estado FIN_WAIT_1 enviando un segmento FIN.

Desde aquí puede pasar a uno de los tres estados que describen cómo se puede realizar un cierre activo dependiendo de cómo se cierre la conexión:

- Con un cierre simultáneo (*simultaneous close*), pasa a CLOSING.
- Con la recepción de un ACK, pasa a FIN_WAIT_2, donde espera recibir un FIN.
- Con la recepción de un FIN y un ACK, pasa a TIME_WAIT, donde espera dos veces el MSL antes de liberar el puerto.

ANOTACIONES

Nota

Consultad el MSL en el apartado 15.3 de esta unidad.

Ya hemos visto que el extremo TCP que recibe un FIN lleva a cabo un cierre pasivo. Por tanto, pasa del estado ESTABLISHED al estado CLOSE_WAIT enviando los indicadores ACK y FIN correspondientes para acabar la conexión.

La fase de establecimiento también se puede seguir con facilidad por medio del diagrama de estados, teniendo en cuenta qué extremo efectúa el cierre activo y cuál lleva a cabo el cierre pasivo.

Actividades

- Utilizad el comando `netstat` para ver el estado de las conexiones TCP que tengáis en este momento. Si no tenéis ninguna aplicación en la red, conectaos a algún servidor con la web, haced un ftp o un telnet a alguna máquina.

- Suponed una interacción entre un cliente y un servidor en la que el cliente hace una apertura activa y el servidor una apertura pasiva, a continuación se intercambian un segmento de datos, con sus correspondientes reconocimientos, y finalizan con un cierre activo por parte del servidor, pasivo por parte del cliente. Dibujad el diagrama de tiempo que muestre el intercambio de segmentos, y en paralelo, los estados donde se encuentran las dos entidades en cada momento.

14.6. Transferencia de la información

Una vez establecida la conexión, el TCP puede empezar la transferencia de segmentos TCP en ambos sentidos. Para tansmitir información de manera fiable, el TCP implementa protocolos de control de errores y de flujo. Los pasos que sigue el TCP para transferir la información son los siguientes:

1) Cuando el TCP envía datos, mantiene un **temporizador (*timeout*)** hasta que recibe un reconocimiento (ACK) del receptor. Si el temporizador salta, el TCP retransmite los datos.

2) Cuando el TCP recibe un segmento de datos, envía un reconocimiento. Este último se puede retornar retrasado (no de inmediato) si el TCP lo considera necesario.

3) Si un segmento recibido es incorrecto (el *checksum* lo indica), el TCP descarta el segmento y no debería enviar la información. De hecho, como el TCP utiliza la técnica de *piggybacking* (aprovecha los segmentos de datos que viajan en sentido contrario), lo que hace es retornar un segmento con el mismo número de ACK que había reconocido la última vez. El transmisor verá un ACK con un número repetido e interpretará que no le reconocen la información. Este número se denomina *ACK duplicado*.

En caso de que no tuviera datos para enviar en sentido contrario, el TCP puede enviar un segmento que no contenga información (con un campo de datos de cero bytes). Este segmento tendría el indicador ACK activado y reconocería los bytes pertinentes en el campo *Número de ACK*. El número de secuencia no se habría incrementado, puesto que no envía datos.

4) Si los segmentos llegan desordenados (por debajo hay el IP, que está no orientado a la conexión), el TCP reordena los segmentos y pasa los datos (bytes) correctamente ordenados a la aplicación. Si recibe segmentos duplicados, el TCP descarta las copias.

5) Como el TCP posee una memoria limitada, debe efectuar un control de flujo. Con esta finalidad, cada extremo avisa de los datos que está dispuesto a recibir en cada momento utilizando el campo Ventana (se trata de un mecanismo de ventana deslizante basado en bytes que explicaremos más adelante).

El tipo de información que es preciso enviar puede dividirse en datos interactivos y datos de gran volumen o *bulk data*. La diferencia entre ellos radica en la cantidad de información que se transmite. Los datos interactivos transmiten pocos bytes de información (alrededor de 10), mientras que los datos de gran volumen transmiten gran cantidad de datos (ocupan la totalidad del tamaño del segmento TCP). Conviene considerar que no es lo mismo cargar la red con paquetes pequeños de información que con paquetes grandes. El TCP puede aplicar en cada caso técnicas diferentes de manera automática, para aprovechar la red al máximo.

Ejemplo

- Datos interactivos: los que transmiten aplicaciones tales como `telnet` o `rlogin`.
- *Bulk data*: los que transmiten aplicaciones como correo electrónico o `ftp`.

14.6.1. Transmisión de datos interactivos

En este tipo de comunicación, es normal enviar pocos datos. En una aplicación del tipo `telnet`, por ejemplo, un usuario cliente podría ejecutar el comando de Unix `ls` y obtener un listado de un directorio por parte del servidor. En esta transferencia de información intervienen pocos bytes desde el origen (cliente) hasta el destino (servidor) y se utilizan conjuntamente dos técnicas para obtener un mejor aprovechamiento de la red:

- Reconocimientos retrasados.
- Algoritmo de Nagle.

Reconocimientos retrasados

En este tipo de transferencia, es normal que el TCP no envíe los reconocimientos ACK inmediatamente después de recibir los datos, sino que esté un tiempo esperando a que haya datos para enviar en sentido contrario. De este modo, puede utilizar la técnica *piggybacking* y enviar el reconocimiento encapsulado en los datos que retornan al cliente.

Es posible que el servidor se ahorre enviar un segmento que sólo reconoce, pero que no contiene datos. Es típico que el TCP espere (utiliza un temporizador) unos 200 ms por si hay datos para transmitir antes de enviar el ACK. Una vez ha transcurrido este tiempo, el TCP reconoce los datos recibidos hasta el momento con un segmento de datos, si dispone de datos para enviar en sentido contrario (*piggybacking*), o con un segmento sin datos (el número de secuencia no varía). En cualquiera de los dos casos, el indicador ACK estará activado y el número ACK reconocerá los datos pertinentes.

Algoritmo de Nagle

En numerosas ocasiones un cliente tiene muy pocos datos para enviar (por ejemplo, sólo 1 byte). En este caso el TCP enviaría un segmento sólo con 1 byte de datos y con 20 bytes de cabecera TCP. El IP añadiría 20 bytes más de cabecera, lo que proporciona un total de 40 bytes de control y 1 de datos. Si se transmiten muchos segmentos de

ANOTACIONES

este tipo, la eficiencia es muy baja. Una solución a esta baja eficiencia de transmisión es aplicar el algoritmo de Nagle.

Utilizando el algoritmo de Nagle, una conexión TCP sólo puede tener un segmento de tamaño pequeño (pocos bytes) sin que se haya reconocido; es decir, sólo puede haber un único segmento de tamaño pequeño viajando por la red (en vuelo). El resto de los segmentos de tamaño pequeño no se pueden transmitir hasta que el ACK del segmento pequeño que esté viajando por la red haya llegado.

Así, los segmentos que están esperando para ser transmitidos se almacenan hasta que se recibe el ACK del segmento en vuelo. Cuando este último llega, la conexión TCP puede enviar un segmento que contenga todos los datos almacenados hasta este momento, formando un segmento mayor.

El algoritmo de Nagle funciona cuando los retardos en la red son grandes; es decir, si la conexión cruza una WAN. En caso de que la conexión sea local, en una LAN, es difícil que se aplique este algoritmo a causa de la alta velocidad de la red.

Nota

En la librería de *sockets* API, el indicador que desinhibe el algoritmo de Nagle es el TCP_NODELAY.

En ocasiones, es interesante desinhibir el algoritmo de Nagle, puesto que la aplicación no puede esperar. El movimiento del ratón en *X Windows System* provoca segmentos pequeños. Estos movimientos del ratón deben entregarse sin retardos para que el usuario interactivo no lo note. Las librerías de *sockets* deben permitir, activando indicadores, desinhibir el algoritmo de Nagle.

14.6.2. Transmisión de datos de gran volumen. Control de flujo por ventana deslizante

En las comunicaciones en que se envía una ingente cantidad de datos de gran volumen (correo electrónico, transferencias FTP, etc.), como las memorias intermedias de recepción se pueden llenar, es necesario un protocolo de ventana deslizante (*sliding window*) para controlar el flujo de datos, con la diferencia, respecto de los protocolos del nivel de enlace, de que en el TCP la ventana de transmisión es variable.

La idea es que cada extremo TCP regula la cantidad de datos que el otro extremo puede transmitir. Con esta finalidad, cada extremo TCP notifica al extremo opuesto, cada vez que envía un segmento, la ventana que puede aceptar en este momento. El TCP opuesto actualiza su ventana de transmisión de acuerdo con este valor.

Mientras que el TCP transmisor marca los bytes que ha transmitido con un número de secuencia, el TCP receptor retoma los bytes que recibe y los reconoce con un ACK. Los reconocimientos ACK especifican siempre el número de secuencia del próximo octeto que el receptor espera recibir.



En el TCP se reconocen posiciones de bytes en el flujo de datos hasta la última posición que ha recibido correctamente, sin tener en cuenta el segmento al que pertenecen.

El TCP sólo activa un temporizador de retransmisiones que reprograma cuando recibe un reconocimiento o cuando salta el temporizador. Más adelante veremos cómo el TCP programa el temporizador de retransmisiones. La cabecera del segmento TCP especifica tres parámetros esenciales en el funcionamiento del protocolo de ventana deslizante:

- El **número de secuencia**, que indica a su conexión opuesta el primer byte de datos que contiene el segmento transmitido.
- El **número de reconocimiento (número ACK)**, que indica a su conexión opuesta el próximo byte que espera recibir y, por tanto, el último byte recibido correctamente.
- La **ventana**, que indica a su conexión opuesta el tamaño de la memoria intermedia de recepción y, por tanto, el tamaño de la ventana que el transmisor debe utilizar.

Actividad

Asumimos que un extremo cliente TCP ha elegido el 28.325 como número de secuencia inicial (ISN), mientras que el extremo servidor TCP ha elegido como ISN el 12.555. ¿Qué indica un segmento cliente TCP con número de secuencia 29.201, número ACK 12.655 y ventana 1.024?

Nota

Recordad que el TCP es bidireccional y que un segmento TCP reconoce, por medio de *piggybacking*, los datos que recibe con un ACK que debe estar numerado.

ANOTACIONES

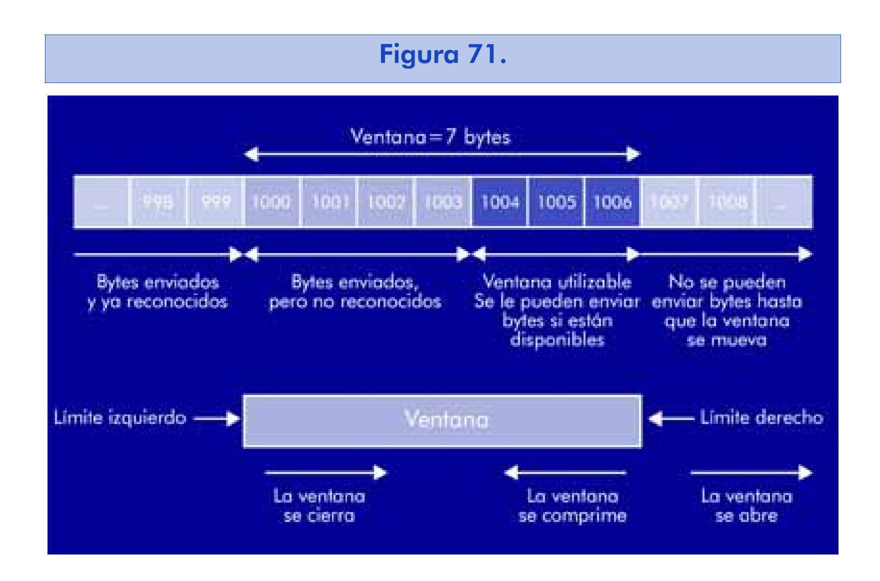
Solución

El número de secuencia indica que el cliente ya ha transmitido desde el byte 28.325 hasta el byte 29.200 (875 bytes en total) y que en este segmento transmitirá a partir del byte 29.201. El número ACK indicará al servidor que el cliente ha recibido correctamente hasta el byte 12.654 y que espera recibir a partir del 12.655. La ventana indica al servidor que el cliente sólo puede aceptar 1.024 bytes antes de confirmarlos. Por consiguiente, el servidor TCP actualizará su ventana de transmisión a 1.024.

Con el objetivo de estudiar el mecanismo de ventana deslizante, analizaremos un caso sencillo. Asumiremos que ya se ha establecido la conexión y se ha asignado la ISN para ambos extremos.

En la figura siguiente, podemos ver como funcionaría el protocolo de ventana deslizante para el TCP transmisor. El TCP receptor le ha indicado que esté dispuesto a recibir 7 bytes. Por tanto, la ventana de transmisión de TCP transmisor es de 7 bytes:

Figura 71.



Podemos interpretar la ventana deslizante de la manera siguiente:

1) El TCP ha enviado bytes hasta el número de secuencia 1.003. De estos bytes, el TCP receptor le ha reconocido hasta el 999; faltan por reconocerle los bytes 1.000 a 1.003.

2) Como la ventana de transmisión es de 7 bytes y ya ha transmitido 4, el TCP todavía puede transmitir 3 bytes antes de agotarla (bytes 1.004 a 1.006).

3) El TCP sólo podrá transmitir del byte 1.007 en adelante en los casos siguientes:

- Si el TCP receptor le reconoce los bytes a partir del 1.000, de manera que el límite izquierdo de la ventana se moverá hacia la derecha.
- Si el TCP receptor le advierte de una ventana superior a 7, de manera que el límite derecho de la ventana se moverá hacia la derecha.
- Una combinación de las dos soluciones anteriores.

Como podéis observar, el TCP receptor puede advertir una nueva ventana de transmisión. Cada vez que reconozca datos, avisará de la nueva ventana que está dispuesta a recibir. El TCP transmisor actualizará esta última.



La ventana puede experimentar tres tipos de movimiento:

1) La **ventana se cierra** al moverse el límite izquierdo hacia la derecha cuando los datos enviados son reconocidos.

2) La **ventana se abre** al moverse el límite derecho hacia la derecha y permite que el TCP envíe más datos. Esta apertura tiene lugar cuando el receptor libera espacio de su memoria y puede advertir una nueva ventana.

3) La **ventana se comprime** cuando el límite derecho se mueve hacia la izquierda.

Algunos puntos que podemos resumir de la figura de la ventana deslizante son los siguientes:

- Si el límite izquierdo alcanza el límite derecho, se dice que la ventana vale cero (*zero window*). Ello hace que el transmisor detenga el envío de datos.

ANOTACIONES

- Se recomienda que el TCP transmisor no comprima la ventana de transmisión.

- Es preciso que distingamos el hecho de que la ventana se comprima (el límite derecho se mueve hacia la izquierda) del hecho de que la ventana disminuya de tamaño (se advierte una ventana más pequeña, pero el límite derecho no se mueve hacia la izquierda).

Nota

Supongamos una ventana de 7 bytes como en la figura-de la ventana deslizante. El receptor reconoce los bytes 1.000 a 1.003 y advierte una ventana de 5 bytes. Como podéis deducir, el límite izquierdo vale ahora 1.004; el límite derecho, 1.008 (se ha movido hacia la derecha), y la nueva ventana, 5. En este caso, la ventana de recepción debe reducirse, pero no se ha comprimido.

En cambio, si el receptor sólo reconoce 1 byte (el byte1.000) y advierte una ventana de 1 byte, el transmisor se encontrará con un problema. Una ventana de 1 byte significa que sólo podía haber transmitido 1 (el 1.001), pero ya había transmitido 3, incluyendo el reconocido (del 1.000 al 1.003). Por tanto, el receptor debe asegurarse de advertir al menos tantos bytes como el transmisor le puede haber enviado con la ventana anterior. Si sólo reconoce 1 byte, la ventana advertida debe ser de 6 bytes; si reconoce los 4 bytes, esta última debe ser, al menos, de 3 bytes, puesto que el transmisor ya los podría haber transmitido.

Ejemplo

Utilizaremos el programa `tcpdump` para observar cómo funciona el protocolo de ventana deslizante. Asumimos que hemos efectuado un `rlogin` de argos a *helios* (*argos* % *`rlogin`* *helios*) y ya estamos conectados a *helios*. Una vez nos encontramos en *helios*, ejecutamos el comando `ls`. Este último retorna por salida estándar el listado de directorios del directorio del usuario (*home directory*) en *helios* que ocupan 811 caracteres (representa el envío de 811 bytes)

```
helios % ls
```

ANOTACIONES

Las líneas que obtenemos con el programa `tcpdump` (numeradas del 1 al 13) son las siguientes:

```
1)15:56:59.506091 argos.1023 > helios.login: P 37:38 (1)ack 596 win 31744
2)15:56:59.516091 helios.login > argos.1023: P 596:597 (1) ack 38 win 8760
3)15:56:.59.526091 argos.1023 > helios.login: .ack 597 win 31744
4)15:56:59.846091 argos.1023 > helios.login: P 38:39 (1)ack 597 win 31744
5)15:56:59.856091 helios.login > argos.1023: : P 597:600 (3) ack 39 win 8760
6)15:56:59.866091 argos.1023 > helios.login: .ack 600 win 31744
7)15:57:00.116091 argos.1023 > helios.login: P 39:40 (1)ack 600 win 31744
8)15:57:00.126091 helios.login > argos.1023: P 600:603 (3) ack 40 win 8760
9)15:57:00.136091 argos.1023 > helios.login: .ack 603 win 31744
10)15:57:00.146091 helios.login > argos.1023: P 603:658 (55) ack 40 win 8760
11)15:57:00.156091 argos.1023 > helios.login: .ack 658 win 31744
12)15:57:00.166091 helios.login > argos.1023: P 658:1414 (756) ack 40 win 8760
13)15:57:00.176091 argos.1023 > helios.login: .ack 1414 win 31744
```

La interpretación de estas líneas es la siguiente: *argos* ya ha enviado 36 bytes, mientras que *helios* ya ha enviado 595 (información que ambos han intercambiado desde el principio de la conexión, como pueden ser logins, usernames, etc.). Deducimos esta información de la primera línea del ejemplo.

1) *argos* envía el carácter 'l'. El indicador P señala PUSH. El número de secuencia avanza de 37 a 38.

2) *helios* retorna un eco del carácter 'l'. Su número de secuencia avanza de 596 a 597 y reconoce el byte recibido (ACK = 37 + 1 = 38).

3) *argos* reconoce el eco: ACK = 597 + 1 = 598.

4) *argos* envía el carácter 's'. El número de secuencia avanza de 38 a 39. El ACK no reconoce nada porque vale igual que antes: ACK = 597.

5) *helios* hace un eco que ocupa 3 bytes (BS + 1 + s). El número de secuencia avanza tres posiciones (de 597 a 600) y reconoce el carácter 's' , puesto que ACK = 38 + 1 = 39.

6) *argos* reconoce el eco con un ACK = 600.

7) *argos* envía el retorno de carro (CR). El número de secuencia avanza una posición.

8) *helios* hace un eco del CR y, asimismo, retorna otro CR seguido de un LF. Ello significa el envío de 3 bytes. Reconoce el CR, puesto que ACK = 40.

Nota

Recordad que PUSH indica al receptor que pase los datos inmediatamente a la aplicación; es decir, que no los deje durante un tiempo en la memoria intermedia de recepción.

ANOTACIONES

9) *argos* reconoce estos tres caracteres.

10) *helios* responde a 'ls' enviando 55 de los 811 bytes que debe enviar. El número de secuencia avanza de 603 a 658. El ACK se mantiene a 40.

11) *argos* reconoce estos 55 bytes enviando un ACK de 659.

12) *helios* transmite el resto de los 811 bytes, es decir, 756 bytes.

13) *argos* reconoce estos bytes avanzando el ACK a 1.414.

Como podemos observar en este ejemplo, el TCP divide la información que hay que enviar en dos segmentos: un segmento de 55 bytes y otro de 756 bytes. Conviene remarcar que rlogin envía los comandos carácter a carácter y que, además, la aplicación remota hace un eco de estos caracteres. Por ello, en las primeras líneas se envía primero la 'l', después la 's', a continuación el retorno de carro, etc. Lo que nos interesa de este ejemplo es ver cómo avanzan las ventanas al emitir y al reconocer bytes. Por tanto, no justificaremos por qué rlogin retorna ecos, ni por qué añade un carácter LF al retorno de carro.

14.6.3. Temporizadores y retransmisiones

El TCP activa hasta cuatro temporizadores de diferente tipo para conseguir una entrega fiable de la información. En este subapartado nos centraremos únicamente en el **temporizador de retransmisiones** o **RTO** (del inglés, *retransmission time out*).

Conviene recordar que tanto los segmentos como los reconocimientos se pueden perder durante la transmisión, de manera que es preciso utilizar un temporizador de retransmisiones.

Como ya hemos mencionado, el temporizador se programa cada vez que se recibe un reconocimiento, o bien cuando salta porque el reconocimiento no ha llegado a tiempo (o, simplemente, no ha llegado).

ANOTACIONES

Definimos el **tiempo de ida y vuelta** o **RTT** (del inglés, *round trip time*) como el tiempo que transcurre desde que se transmite un segmento, hasta que es reconocido (el ACK vuelve al transmisor). El RTT se puede medir restando el instante en que el TCP transmisor emite el segmento y el instante en que recibe el ACK.

Lo más lógico sería activar el temporizador de retransmisión en el tiempo de ida y vuelta (RTO = RTT). Sin embargo, es evidente que los retardos que experimentan los segmentos son variables: si se activa el RTO en el RTT que ha experimentado el segmento anterior, no se puede asegurar que este segmento no tenga un retardo superior y que el temporizador no salte antes de tiempo. Por otra parte, si el temporizador se programa a un valor mucho mayor que RR, cada vez que haya una pérdida estaremos esperando en balde un buen rato, con la consiguiente pérdida de eficiencia.

Existen diferentes alternativas para programar el temporizador, todas orientadas a encontrar un valor adecuado de RTT; es decir, que no sea demasiado corto ni demasiado largo.

Lectura complementaria

Podéis encontrar los diferentes algoritmos propuestos para el cáculo de RTT en la obra siguiente:

W.R. Stevens (1994) "The protocols". *TCP/IP Illustrated* (vol. 1) Willmington: Addison-Wesley.

ANOTACIONES

IV. Aplicaciones Internet

15. El modelo cliente/servidor

Las redes de computadores han hecho aparecer un concepto nuevo en el mundo de la programación: la **programación distribuida**. Con esta última se pretende aprovechar la potencia y los recursos de los ordenadores interconectados para llevar a cabo una tarea de forma cooperativa.

Una aplicación distribuida está formada por varios programas que se ejecutan en ordenadores diferentes y que se comunican por medio de la red que une a los ordenadores.



Conviene destacar que cada programa, por sí solo, no puede hacer nada. Es necesaria la **colaboración** de todos para que la aplicación en conjunto produzca resultados útiles.

La cooperación de los diferentes trozos de código que forman la aplicación debe seguir un **protocolo**. Este último se puede elaborar a medida para cada aplicación que se desarrolle, o se puede definir un estándar que todo el mundo pueda seguir y así ahorrarse, de este modo, los diseños particulares. Asimismo, la existencia de un protocolo estándar garantiza la posibilidad de interactuar con productos de diferentes fabricantes.

Existen varios estándares de aplicaciones distribuidas; sin embargo, sin lugar a dudas, el que ha tenido más éxito es el que sigue el modelo cliente/servidor.



En el modelo cliente/servidor, la aplicación se divide en dos partes, con dos roles claramente diferenciados:

Servidor: ofrece un servicio que puede ser el acceso a

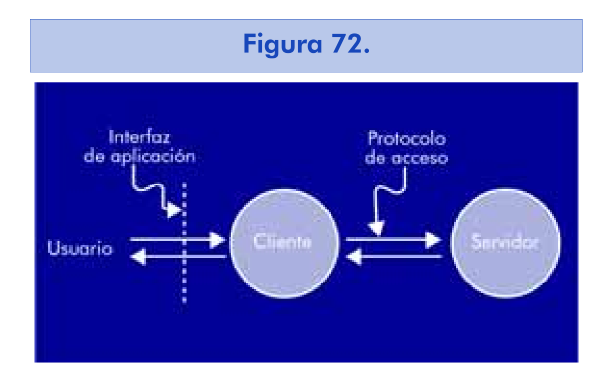
ANOTACIONES

un recurso, la ejecución de operaciones matemáticas complejas, procesamiento de datos, etc.

Cliente: realiza una petición al servidor y espera un resultado de la misma.

Por norma general (al menos es lo que dice el modelo), los usuarios acceden a la aplicación por medio del cliente.

Figura 72.

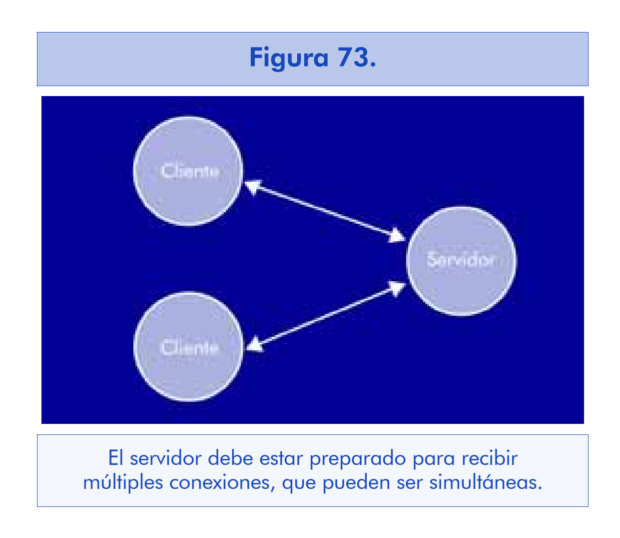


El usuario de las aplicaciones puede ser una persona, pero también otra aplicación. En el primer caso, el cliente debe disponer de una **interfaz de usuario** que permita el diálogo, canalizando las peticiones y mostrándole los resultados. En el segundo, el acceso se suele implementar con llamadas a funciones.

El modelo se puede complicar si se añaden más entidades a la aplicación distribuida. Veamos dos ejemplos:

- Múltiples clientes:

Figura 73.

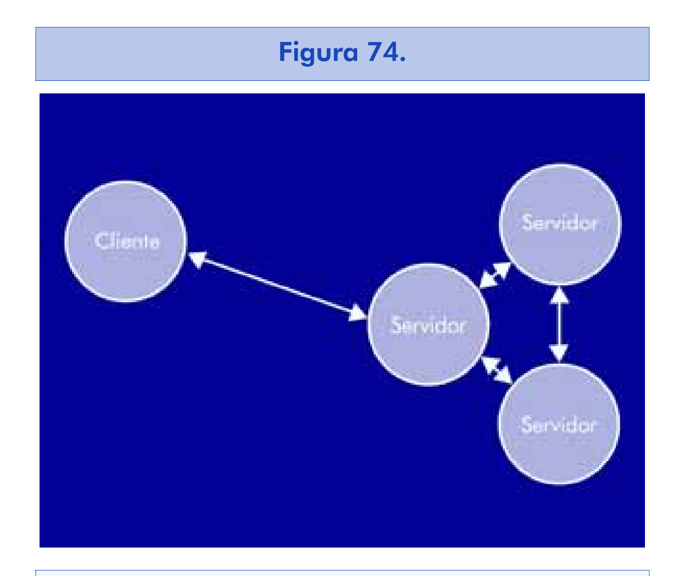


El servidor debe estar preparado para recibir múltiples conexiones, que pueden ser simultáneas.

ANOTACIONES

- Múltiples servidores:

Figura 74.



Varios servidores ofrecen servicios complementarios. Si el servidor no puede satisfacer la petición del cliente, pero está en contacto con otro servidor que sí que puede hacerlo, entonces se lo solicita.
Los servidores pueden dialogar entre sí con protocolos propios o siguiendo el modelo cliente/servidor.

Desde el punto de vista del cliente, esta multiplicidad de servidores es completamente transparente: realiza una petición a un servidor y espera una respuesta del mismo.

Cuando diferentes servidores cooperan para ofrecer un servicio, hablamos de ***sistema servidor***.

La gran mayoría de las aplicaciones que se utilizan en Internet siguen este modelo cliente/servidor. De aquí los nombres que se emplean con asiduidad: servidor web, cliente de correo, servicio de nombres, etc.



El diseño de una aplicación distribuida que siga el modelo cliente/servidor incluye básicamente dos elementos: **la especificación de los servicios** que ofrece el servidor y **la especificación del protocolo de acceso**, donde se describe cómo se piden estos servicios y cómo debe retornarse el resultado.

ANOTACIONES

15.1. El modelo *peer-to-peer*

El modelo cliente/servidor implica una cierta asimetría. Tendemos a ver los servidores como sistemas potentes que ofrecen sus servicios a máquinas mucho más sencillas, los clientes. El gran desarrollo de Internet en su vertiente comercial ha contribuido a esta visión. Los usuarios de Internet, desde sus PC, tienen un claro comportamiento de cliente, accediendo a páginas web alojadas en servidores remotos, o enviando correos electrónicos a otros usuarios.

A medida que el uso de Internet ha madurado, los usuarios de la Red han sentido la necesidad de desempeñar un papel más activo, y convertir sus máquinas en servidores que otros puedan aprovechar, o desarrollar nuevos modelos de aplicación donde los roles no están tan claramente diferenciados. Uno de estos modelos es el que se conoce como *peer-to-peer* ('de igual a igual'), y las aplicaciones más típicas que siguen este modelo de comportamiento son los sistemas de compartición de archivos o la mensajería instantánea.

Nota

Diferentes características de la Red que se han ido añadiendo por cuestiones de seguridad o de eficiencia de la Red están perjudicando mucho el desarrollo de entornos peer-to-peer. Podemos citar como las más paradigmáticas los cortafuegos, las direcciones IP dinámicas, el NAT (*Network Address Translation*) y el ancho de banda asimétrico que ofrecen en muchos casos los proveedores de Internet en los dos sentidos de transmisión. La mayoría de aplicaciones intenta sobreponerse a estos obstáculos con estrategias más o menos afortunadas. De todas formas, está claro que en un futuro tienen que cambiar cosas en el nivel de red para no limitar las enormes posibilidades que tenemos en el nivel de aplicación.

De hecho, Internet en su origen era *peer-to-peer*. Los primeros ordenadores conecatodos a la Red eran parecidos, y una aplicación como `ftp` permitía la transferencia de archivos entre cualesquiera de estas máquinas. Dicho de otra forma, que el protocolo que usa la aplicación siga el modelo cliente/servidor no es incompatible con que la red esté formada por ordenadores de potencia parecida y entre ellos se vean como una red *peer-to-peer*.

ANOTACIONES

Nota

El servicio DNS y el servicio News (o Usenet o NNTP), presentes en Internet desde sus inicios son claramente *peer-to-peer*.

Porque en el fondo, si analizamos como funcionan las aplicaciones típicas de entornos *peer-to-peer*, como el extinto Napster, o Gnutella o ICQ o Jabber, veríamos que siguen protocolos típicos cliente/servidor, tipo "yo te pido, tu me das". Lo que pasa es que en este tipo de entornos cualquiera puede hacer a la vez de cliente y de servidor. Por tanto, es erróneo considerar ambos modelos excluyentes.



Un sistema puede ser *peer-to-peer* si presenta una cierta simetría en el sentido que las diferentes máquinas que lo componen son parecidas y pueden desempeñar los mismos roles, y usar un modelo cliente/servidor en el desarrollo de las aplicaciones.

Por otra parte, existen aplicaciones consideradas *peer-to-peer* porque permiten la comunicación directa entre iguales, pero que necesitan de un servidor externo (o varios) donde se centraliza el control de la red (Napster era un claro ejemplo de este modelo). Hay autores que son reacios a considerar estos entornos realmente *peer-to-peer*, y prefieren guardar el término para aquellas redes donde sólo intervienen las máquinas propiedad de los propios usuarios. No es objetivo de este material didáctico entrar en esta discusión.

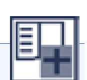
ANOTACIONES

16. Servicio de nombres Internet

La red Internet permite el acceso a una ingente cantidad de ordenadores y, en general, de recursos, la mayoría de los cuales se pueden referenciar mediante un nombre. Por motivos de eficiencia y simplicidad (poder saber con facilidad qué recurso corresponde a cada nombre, poder asignar otros nuevos sin peligro de duplicidades o ambigüedades, etc.), todos los nombres de la red están organizados de una manera jerárquica, formando un **sistema de dominios**.

Para obtener información referida a cualquier nombre, se utiliza un servicio que, funcionalmente, es como una base de datos: se le hacen consultas, que pueden incluir una serie de criterios de selección, y responde con la información solicitada. En un inicio, cuando el número de ordenadores conectados a la red era relativamente pequeño, esta base de datos era gestionada por una única autoridad central, el Network Information Center (NIC).

A medida que se expandía la Red, este modelo de gestión se iba haciendo cada vez más inviable, tanto por el enorme tráfico que generaban las consultas, como por la dificultad de mantener la información actualizada. Fue entonces cuando nació el **servicio de nombres de dominio** (DNS), que sigue el modelo de una base de datos distribuida descentralizada.



Si queréis más información sobre el DNS, podéis consultar las obras siguientes:

P.V. Mockapetris (1987,1 de noviembre). *RFC 1034-Domain names - concepts and facilities*.

P.V. Mockapetris (1987, 1 de noviembre). *RFC 1035-Domain names - implementation and specification*.

ANOTACIONES

La especificación del DNS está contenida en los documentos RFC 1034, que define los conceptos y la organización del sistema de dominios, y RFC 1035, que define el protocolo de acceso al servicio.

16.1. El sistema de nombres de dominio

El sistema de nombres de dominio, en que se basa el DNS, proporciona un espacio de nombres para referenciar recursos, que por norma general son ordenadores conectados a la red, pero que también pueden ser, por ejemplo, buzones de correo electrónico.



En el sistema de nombres de dominio, los nombres están organizados jerárquicamente en forma de árbol. Cada nodo de este último tiene una etiqueta que lo distingue de sus nodos "hermanos".

El **nombre de dominio** correspondiente a un nodo se define como la secuencia formada por las etiquetas existentes en el camino entre este nodo y la raíz.

Cada etiqueta constituye una cadena de caracteres de longitud comprendida entre 0 y 63 caracteres. La etiqueta con longitud 0 está reservada para el nodo raíz: ningún otro nodo puede tener una etiqueta vacía.

Nota

A la hora de comparar nombres, las letras mayúsculas y minúsculas se consideran equivalentes. No obstante, para responder las consultas, los servidores DNS deben enviar los nombres tal como están almacenados en la base de datos, respetando las mayúsculas y minúsculas, en previsión del caso de que en alguna versión futura del servicio se consideren diferentes.

El protocolo de acceso al DNS define el formato para representar los nombres de dominio cuando deben enviarse en las consultas y las respuestas, como veremos más adelante. Este formato no impone restricciones en los caracteres que puede haber en una etiqueta; sin embargo, establece una "sintaxis preferida" para facilitar las implementaciones de otros servicios. De acuerdo con esta última, los únicos caracteres válidos en una etiqueta son las letras, mayúsculas o minúsculas, los dígitos decimales y el símbolo "–", con la excepción de que el primer carácter sólo puede ser una letra, y el último no puede ser "–".

ANOTACIONES

Aparte de la representación definida en el protocolo, cuando se desea utilizar una notación textual para expresar un nombre de dominio, se sigue la convención de concatenar las etiquetas que lo forman, separándolas con el carácter ".". En el orden utilizado, las etiquetas van desde el más bajo nivel jerárquico más bajo hasta el más alto. Por tanto, la última etiqueta es la del nodo raíz que, como acabamos de observar, está vacía, de manera que los nombres acaban en ".", como en el ejemplo siguiente:

```
campus.uoc.edu.
```

Notad la presencia del punto final.

Este ejemplo es el que se denomina un nombre de dominio absoluto, pero también pueden utilizarse nombres relativos, referidos a un nodo origen implícito. Así, dentro del dominio `uoc.edu.` podemos utilizar el nombre `tibet` para referirnos al nombre de dominio `tibet.uoc.edu.`. En numerosas ocasiones, el nodo origen de los nombres relativos es la raíz, de manera que el ejemplo anterior también puede escribirse del modo siguiente:

```
campus.uoc.es
```

Nota

En el Reino Unido antes era usual utilizar el orden inverso en la representación textual de los nombres de dominio, por ejemplo: `uk.ac.ic.doc.src` en lugar de `src.doc.ic.ac.uk`; sin embargo, hoy día esta costumbre se encuentra en desuso.

En un inicio, los TLD (*top level domain*, 'dominios de nivel superior'); es decir, los subordinados directamente a la raíz, se utilizaban para agrupar los diferentes tipos de organizaciones a que pertenecían recursos como, por ejemplo, los que presentamos en la tabla siguiente:

Tabla 6.

Tipos de organización	Nombre del dominio
Empresas y organizaciones comerciales	com
Universidades y centros educativos	edu
Organismos gubernamentales	gov
Organizaciones militares	mil
Otros tipos de organizaciones	org

Nota

En estos momentos, el organismo encargado de gestionar los dominios es la ICANN. Podéis consultar su web: http://www.icann.org.

ANOTACIONES

Desde entonces hasta la actualidad, se han ido añadiendo al nivel superior del sistema de nombres los elementos siguientes:

a) Por un lado, nuevas clases de organizaciones, como `net` (centros relacionados con la gestión de la red), `int` (organizaciones internacionales) o, más recientemente, `biz`, `info`, `name`, `pro`, `aero`, `coop` y `museum`.

b) Por otro lado, cada país tiene asignado un nombre de dominio que coincide con el código correspondiente de dos letras especificado en el estándar ISO 3166, por ejemplo: `es` (España), `fr` (Francia), `de` (Alemania), etc. Lo que significa que, en realidad, las diferentes ramas del árbol se pueden encabalgar y, de hecho, hay nombres pertenecientes a diferentes dominios de alto nivel que corresponden a una misma organización.

Asimismo, es posible que un determinado recurso (ordenador, buzón de correo, etc.) tenga asignados varios nombres, que pueden estar en la misma rama del árbol o en ramas completamente independientes. En este caso, se considera que uno de los nombres es el denominado *nombre canónico* y los otros se denominan *alias*.

16.2. Modelo del DNS

Los agentes que intervienen en el DNS son los siguientes:

Nota

Consultad la definición de sistema servidor en la unidad 15.

a) Los **servidores**, que reciben las consultas y envían las respuestas correspondientes. Además del acceso directo a una base de datos local, en la que se encuentra una parte de la base de datos total del sistema de nombres, el servidor tiene acceso indirecto al resto de la base de datos por medio de otros servidores. El conjunto de servidores DNS constituye, pues, un sistema servidor.

b) Los **resolvedores**, que actúan como clientes del servicio. Por norma general, un resolvedor es un programa o una librería que recibe peticiones de las aplicaciones de usuario, las traduce a consultas DNS y extrae de las respuestas la información solicitada. El resolvedor debe tener acceso directo al menos a un servidor DNS.

ANOTACIONES

Nota

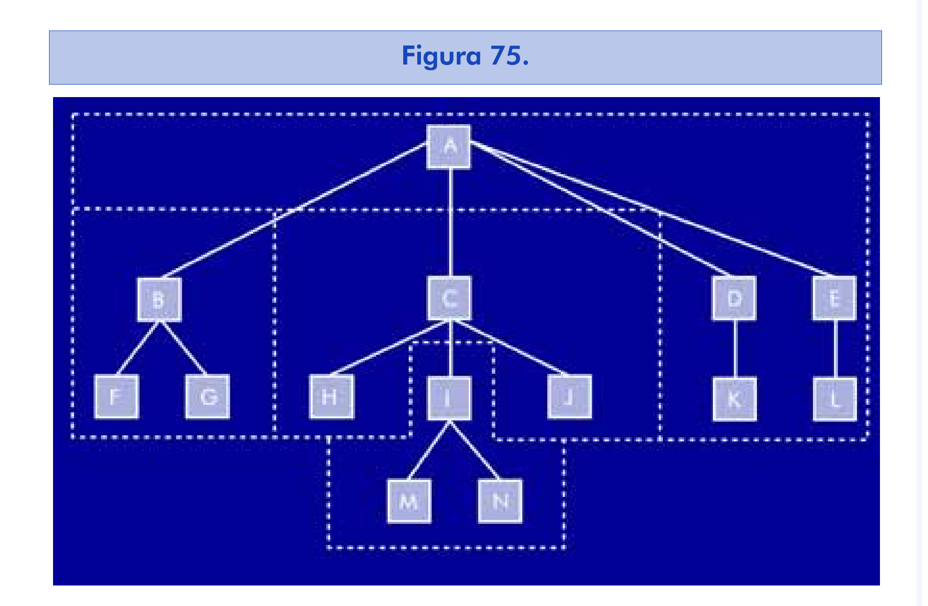
Utilización del resolvedor

Una aplicación que sirva para copiar ficheros almacenados en otros ordenadores puede aceptar que el usuario indique a qué ordenador desea acceder por medio de su nombre. Por consiguiente, para poderse comunicar con él, la aplicación deberá realizar una llamada al resolvedor para obtener su dirección a partir del nombre. Como la aplicación y el resolvedor, por lo general, se encontrarán ubicados en el mismo sistema, no es necesario establecer ningún protocolo de comunicación entre ellos.

Para hacer manejable la gestión y administración del sistema de nombres de dominio, sus nodos están agrupados en zonas. Cada **zona** está formada por un nodo llamado *superior* y todos los que se encuentren en los niveles jerárquicamente inferiores hasta llegar a los nodos terminales (las hojas del árbol) o a nodos de otras zonas que sean superiores.

La figura siguiente muestra un ejemplo de división de un árbol en cuatro zonas:

Figura 75.



El objetivo de agrupar los nodos en zonas consiste en asignar a una autoridad la responsabilidad de gestionar todos los nodos de cada zona.

Nota

Como cada zona se suele denominar con el nombre de dominio de su nodo superior, podemos decir que las zonas de la figura son A, B.A, C.A e I.C.A.

ANOTACIONES

El administrador designado para esta autoridad puede añadir o borrar nodos dentro de su zona, modificar la información de sus nodos o crear nuevas subzonas y delegar su gestión en otras autoridades.

La información referida a una zona debe estar almacenada en la base de datos local de un servidor, del que se dice que es un servidor con autoridad sobre esta zona. Este último puede contestar directamente las consultas que reciba sobre los nodos de su zona, sin necesidad de acceder a otros servidores. Es decir, en este caso, el servidor enviará respuestas con autoridad.

Ejemplo

Una zona puede corresponder a un país y sus subzonas pueden corresponder a organizaciones de este país. Cada organización, a su vez, puede crear subzonas para diferentes departamentos, etc.

Si una consulta se refiere a otra zona, existen dos maneras posibles de generar la respuesta:

- En el **modo no recursivo**, la respuesta únicamente incluye una referencia a algún servidor que puede proporcionar más información. El cliente debe preocuparse de continuar realizando consultas hasta encontrar la respuesta definitiva.

- Si el cliente solicita el **modo recursivo**, es el servidor quien se ocupa de buscar la información donde sea preciso, y sólo puede retornar la respuesta final o un error, pero no referencias a otros servidores.

La especificación DNS establece que todos los servidores deben soportar el modo no recursivo, y que el modo recursivo es opcional, aunque en la práctica, la mayoría de las consultas de los clientes son en modo recursivo.

Cuando un servidor debe responder a una consulta en modo recursivo, pide la información a otros servidores y envía la respuesta recibida a quien ha realizado la consulta. Es habitual que el servidor añada la información obtenida a su base de datos, en lo que se denomina *caché*. De este modo, si recibe otra consulta (del mismo cliente o de otro) en la que se solicita la misma información, no será necesario que se la pida de nuevo a otros servidores, sino que puede aprovechar la ya existente en la caché. Sin embargo, es posible que los datos se hayan modificado en el servidor de origen desde que se solicitaron por primera vez. En este caso, pues, el servidor debe avisar al cliente de que le envía una res-

ANOTACIONES

puesta sin autoridad. Por otro lado, los resolvedores también suelen guardar en una caché propia las respuestas que reciben de los servidores.

Para ofrecer una alta fiabilidad en el servicio, la especificación DNS requiere que para cada zona haya como mínimo dos servidores con autoridad que, por norma general, responden a las siguientes funciones:

a) Uno actúa como **servidor primario** y guarda los ficheros originales de la base de datos correspondiente a la zona; es decir, aquellos que el administrador debe actualizar directamente cada vez que haya una modificación en sus nodos.

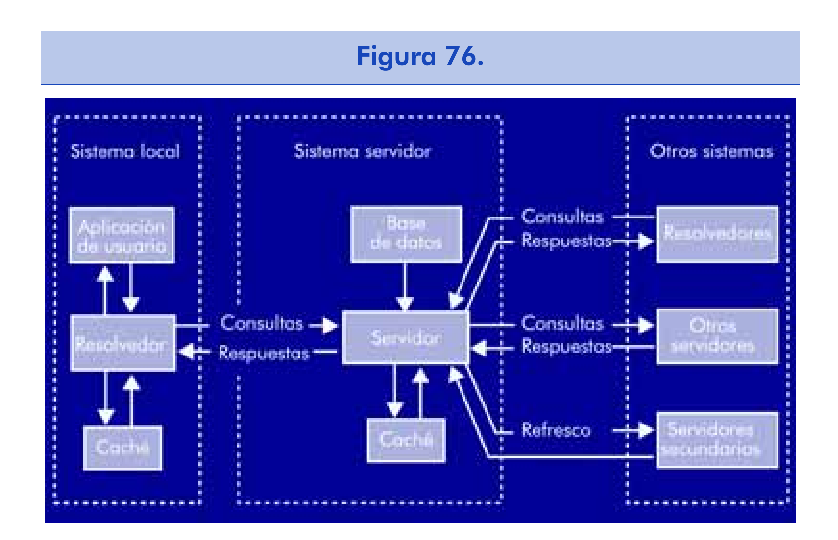
b) Los otros actúan como **servidores secundarios** y actualizan automáticamente sus bases de datos a partir de la del primario; por ejemplo, por medio de consultas periódicas para saber si se ha producido algún cambio. De este modo, si un servidor primario está temporalmente inaccesible (por una caída de la red, del mismo servidor, etc.), los clientes pueden enviar sus consultas a uno de los servidores secundarios.

Nota

En el argot DNS, *refrescar los datos* significa actualizarlos realizando consultas periódicas.

La figura siguiente muestra una posible configuración del servicio DNS en un ordenador:

Figura 76.



En la práctica, se pueden encontrar muchas variantes de este esquema; por ejemplo, que el servidor DNS se encuentre en el mismo ordenador que el resolvedor y que ambos compartan la caché, que el resolvedor tenga acceso a diferentes servidores DNS, etc.

ANOTACIONES

16.3. Base de datos DNS: los registros de recurso

Como hemos visto en los subapartados anteriores, un nombre del sistema de dominios identifica un nodo, y para cada nodo puede haber cierta información almacenada en los servidores correspondientes. El objetivo del servicio DNS consiste en permitir la obtención de dicha información a partir del nombre de dominio.



La información asociada a un nodo consta de un conjunto de registros de recurso. Los registros de recurso de todos los nodos forman la **base de datos DNS**.

Cada registro de recurso consta de los campos siguientes, que se explican a continuación:

- Nombre.
- Tipo.
- Clase.
- Tiempo de vida.
- Datos del recurso.

a) **Nombre**. Contiene el nombre de dominio del nodo al que está asociado el registro.

b) **Tipo**. Indica qué tipo de información contiene el registro. Los valores que podemos encontrar en este campo y los tipos de información que representan son los siguientes:

- `A`: dirección de un ordenador.
- `CNAME`: nombre canónico de un alias.
- `HINFO`: información sobre el tipo de ordenador.
- `MX` (*Mail eXchanger*): nombre de un servidor de correo electrónico para un dominio.
- `NS`: nombre de un servidor DNS con autoridad para una zona.
- `PTR`: nombre de un dominio que contiene información relacionada con un nodo.
- `SOA` (*Start Of Authority*): información sobre el nodo superior de una zona.

ANOTACIONES

Otros valores posibles del campo Tipo son los siguientes:

- MB: nombre de un buzón de correo.
- MG: miembro de un grupo de correo.
- MR: nombre nuevo de un buzón de correo.
- MINFO: información sobre un buzón o una lista de distribución de correo.
- WKS (*Well Known Services*): lista de servicios que proporciona un ordenador.
- TXT: texto descriptivo.
- NULL: registro vacío.

c) **Clase**. Indica la familia de protocolos utilizada en el espacio de nombres. Por norma general, será la familia de protocolos Internet, pero puede haber otros.

d) **Tiempo de vida** (**TTL**). Indica el tiempo máximo que un servidor o un resolvedor puede guardar el registro en su caché.

e) **Datos del recurso** (**RDATA**). El valor de este campo depende del tipo de registro:

- Si el registro es de tipo A, en la clase Internet, el valor es un número de 32 bits que corresponde a una dirección IP.
- Si el registro es de tipo CNAME, el valor es un nombre de dominio que corresponde al nombre canónico del alias asociado con el registro. Si un nodo del árbol de nombres es un alias, la única información que puede tener asociada es un registro CNAME.
- Si el registro es de tipo HINFO, el valor es una cadena de caracteres.
- Si el registro es de tipo MX, el valor tiene dos subcampos, el primero es un número que representa una preferencia (cuanto menor es el número, más preferencia) y el segundo es el nombre de un ordenador que está dispuesto a aceptar mensajes destinados al dominio correspondiente al registro.

Nota

El sistema de nombres de dominio proporciona un conjunto de espacios de nombres, uno para cada clase, todos sobre un mismo árbol. Los datos correspondientes a un nodo en una clase pueden ser diferentes de las del mismo nodo en otra clase (un nodo puede no tener ningún registro de recurso de alguna clase), puesto que las bases de datos (como las divisiones en zonas) son separadas e independientes.

ANOTACIONES

> **Nota**
>
> El estándar RFC 974 especifica cómo deben utilizarse los registros MX para direccionar un mensaje de correo electrónico dada la dirección del destinatario (por ejemplo, `usuari@uoc.edu`). De esta dirección se separa a parte correspondiente al dominio de correo (`uoc.edu`), se consulta el sistema DNS para saber qué servidores aceptan correo para este dominio y se elige uno teniendo en cuenta su preferencia. Si la comunicación con este servidor no tiene éxito, se intenta con los otros por orden de preferencia.
>
> Si no hay ningún registro `MX` asociado al dominio, se entiende que sólo dispone de un servidor de correo: el ordenador que tenga por nombre el del dominio.

- Si el registro es de tipo `NS`, el valor es un nombre de ordenador.
- Si el registro es de tipo `PTR`, el valor es un nombre de dominio.
- Si el registro es de tipo `SOA`, el valor del campo `RDATA` en un registro de este tipo está formado por los siete subcampos siguientes:
  - `MNAME`: nombre del servidor primario de la zona.
  - `RNAME`: nombre de dominio correspondiente al buzón del responsable de la zona.
  - `SERIAL`: número que debe aumentarse cada vez que haya una modificación en los datos de la zona.
  - `REFRESH`: tiempo que debe transcurrir para que los servidores secundarios refresquen sus datos.
  - `RETRY`: tiempo que es preciso esperar para volver a intentar un refresco si no se ha conseguido contactar con el servidor primario.
  - `EXPIRE`: tiempo máximo a partir del cual los datos de un servidor secundario se considerarán sin autoridad si no se han refrescado.
  - `MINIMUM`: valor mínimo del campo `TTL` en los registros de la zona.

> **Nota**
>
> Cada nodo que sea el superior de una zona debe tener asociado un registro SOA.

> **Nota**
>
> El nombre del buzón puede expresarse como un nombre de dominio siguiendo la sintaxis general de separar los nombres de los nodos con "." (en este caso, la parte correspondiente al usuario sería el nodo inferior).
>
> Por ejemplo, en el buzón `usuari@uoc.edu` le corresponde el nombre de dominio `usuari.uoc.edu`.

ANOTACIONES

Nota

Refresco de una zona en un servidor secundario

El proceso de refresco de una zona en un servidor secundario consiste en pedir al primario su registro SOA y comprobar si ha variado el campo SERIAL. Si ha variado, es preciso llevar a cabo una transferencia de los datos de la zona (por ejemplo, con una petición AXFR del protocolo DNS, por medio del protocolo FTP, etc.) y, si no, no es necesario hacer nada.

El protocolo de acceso al DNS, como veremos en el subapartado siguiente, define el formato de representación de los registros de recurso en las respuestas de los servidores. Sin embargo, también es habitual utilizar una representación textual; por ejemplo, en los ficheros de la base de datos que el administrador de una zona puede editar y actualizar. La sintaxis típica de esta representación textual es la siguiente:

```
[nombre] ( [clase] [TTL] | [TTL] [clase] ) tipo RDATA
```

Cada registro se escribe en una línea; sin embargo, si es demasiado largo, se pueden utilizar paréntesis, dentro de los cuales se ignoran los saltos de línea. El símbolo ";" sirve para introducir comentarios en el mismo. El campo Nombre es opcional (por defecto se toma el del registro anterior), como también lo son clase (el valor del campo clase, por norma general, es IN, que correspoonde a la clase Internet) y TTL, que se pueden escribir en cualquier orden. Los campos que representan dominios pueden ser absolutos (acabados en ".") o relativos (respecto a un origen preestablecido), y los que representan tiempo se expresan en segundos.

El campo Nombre puede empezar con la etiqueta "*" para indicar que la información del registro se aplica a los dominios que tengan cualquier etiqueta o secuencia de etiquetas en el lugar del "*" y no figuren en ningún otro registro.

ANOTACIONES

Nota

Líneas del fichero de datos de un servidor acme.com

Éstas podrían ser algunas líneas del fichero de datos de un hipotético servidor de la zona acme.com:

```
acme.com.   IN   SOA   servidor.acme.com.   admin.acme.com. (
                               38       ; SERIAL
                               7200     ; REFRESH
                               600      ; RETRY
                               3600000  ; EXPIRE
                               60 )     ; MINIMUM
                            NS   servidor.acme.com.
                            NS   servidor2.acme.com.
                            NS   dns.competencia.com.
                            MX   10   correo.acme.com.
                            MX   20   servidor.acme.com.
  *.acme.com.               MX   10   correo.acme.com.
  servidor.acme.com.        A    128.52.46.32
                            A    128.32.11.99
  servidor2.acme.com.       A    128.52.46.33
  correo.acme.com.          A    128.52.46.34
```

Un uso habitual de los registros PTR es facilitar la obtención del nombre de un ordenador a partir de su dirección por medio de un dominio especial denominado IN-ADDR.ARPA. El DNS proporciona una operación para efectuar consultas inversas; es decir, dado un registro, retorna el nombre de dominio a que corresponde. Sin embargo, esta operación puede ser muy costosa para el servidor, y no se puede garantizar que la respuesta sea única o completa, puesto que puede haber otros servidores que contengan información relacionada.

No obstante, como la traducción de direcciones IP a nombres es útil e, incluso, necesaria, en algunos protocolos (por ejemplo, rlogin y rsh) se ha adoptado un mecanismo para facilitarla que consiste en introducir registros adicionales que actúan como índice. De este modo, cada vez que se modifica una dirección en la base de datos DNS, o se añade una, es necesario actualizar dos registros:

- El registro A que tiene por nombre el del ordenador.
- Un registro PTR que tiene un nombre perteneciente al dominio IN-ADDR.ARPA.

Los nombres del dominio IN-ADDR.ARPA pueden tener hasta cuatro etiquetas (sin contar IN-ADDR y ARPA), que representan los valores posibles de los bytes de una dirección IP en decimal, entre 0 y 255. De estas etiquetas, la de nivel más alto corresponde al primer

byte de la dirección, y la de nivel más bajo, al último byte. De este modo, se puede tener una estructura de zonas y autoridades delegadas similar a la del resto de los dominios. Un nombre del dominio IN-ADDR.ARPA que posea las cuatro etiquetas numéricas corresponderá a la dirección de un ordenador, y se le asociará un registro PTR que apunte al nombre de este ordenador. Los nombres que dispongan de menos de cuatro etiquetas numéricas serán direcciones de redes y podrán tener asociados registros PTR apuntando al nombre de las pasarelas conectadas a estas redes.

Nota

Registros de traducción

Siguiendo el ejemplo anterior, los registros del dominio IN-ADDR.ARPA siguientes podrían servir para llevar a cabo la traducción de direcciones a nombres:

```
32.46.52.128.IN-ADDR.ARPA.  PTR  servidor.acme.com.
33.46.52.128.IN-ADDR.ARPA.  PTR  servidor2.acme.com.
34.46.52.128.IN-ADDR.ARPA.  PTR  correo.acme.com.
99.11.32.128.IN-ADDR.ARPA.  PTR  servidor.acme.com.
46.52.128.IN-ADDR.ARPA.     NS   servidor.acme.com.
                            NS   servidor2.acme.com.
                            PTR  servidor.acme.com.
11.32.128.IN-ADDR.ARPA.     NS   servidor.acme.com.
                            NS   servidor2.acme.com.
                            PTR  servidor.acme.com.
```

Nota

Los nombres del dominio IN-ADDR.ARPA son una excepción a la sintaxis preferida para las etiquetas, puesto que deben empezar con un dígito.

16.4. Protocolo

16.4.1. Mecanismos de transporte

Para acceder al DNS se pueden utilizar los protocolos de transporte UDP o TCP. Por norma general, se utiliza UDP en las consultas de los clientes, por su simplicidad y por los pocos recursos que requiere. Si no llega la respuesta a un datagrama en un tiempo determinado, simplemente se retransmite (hasta que haya transcurrido un tiempo máximo). En cambio, se utiliza TCP cuando conviene asegurar una transmisión fiable; por ejemplo, en las transferencias de datos de una zona de un servidor a otro. Tanto en un caso como en el otro, el número de puerto utilizado es el 53.

ANOTACIONES

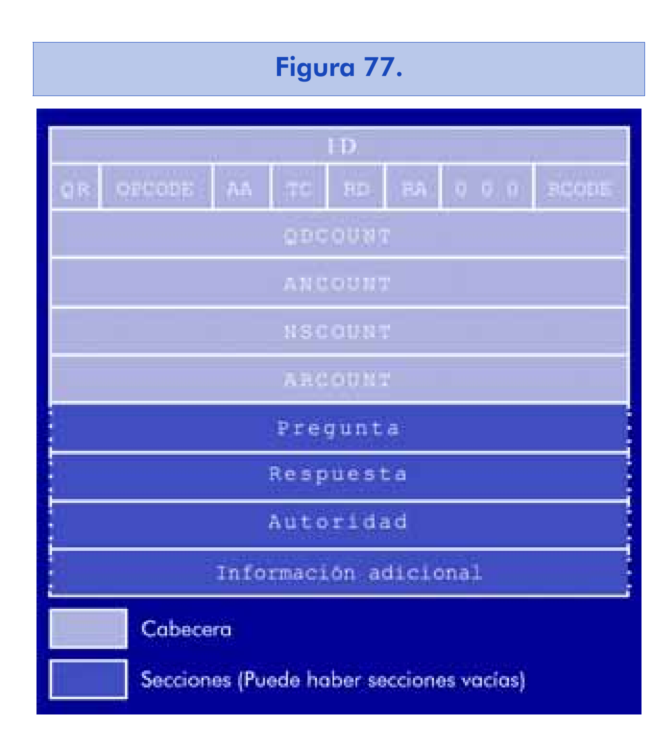
Las consultas y las respuestas se intercambian por medio de **mensajes**, cuyo formato depende del protocolo que se utilice:

- En UDP, cada mensaje se envía en un datagrama, con una longitud máxima de 512 bytes.
- En TCP, los mensajes se envían prefijados con 2 bytes, que indican su longitud, para saber dónde acaban, puesto que lo más normal es intercambiar más de uno utilizando una misma conexión.

16.4.2. Mensajes

El esquema siguiente muestra la estructura de un mensaje DNS:

Figura 77.



Cada mensaje está formado por una cabecera y cuatro secciones (pregunta, respuesta, autoridad e información adicional).

La **cabecera** consta de los campos siguientes:

- `ID` es un número de 16 bits que asigna quién realiza la consulta y que se copia en la respuesta para que se sepa a qué consulta corresponde.

ANOTACIONES

- **QR** (*Query/Response*) es un bit que indica si el mensaje es una consulta (0) o una respuesta (1).
- **OPCODE** es un código de operación de 4 bits. En la actualidad, hay definidos los de consulta directa (QUERY, código 0), consulta inversa (IQUERY, código 1) y petición de estatus (STATUS, código 2). Por norma general, el valor de este campo será QUERY, puesto que la operación IQUERY es opcional y poco soportada, y la operación STATUS no está definida en la especificación RFC 1034.
- **AA** (*Authoritative Answer*) es un bit que informa de que el mensaje es una respuesta con autoridad.
- **TC** (*Truncation*) es un bit que avisa que el mensaje se ha truncado porque no cabe en un datagrama. Para saber cuál es su contenido completo, es preciso utilizar el protocolo TCP.
- **RD** (*Recursion Desired*) es un bit que indica que el cliente solicita respuesta en modo recursivo.
- **RA** (*Recursion Available*) es un bit que informa de que el servidor soporta el modo recursivo. Si los bits RD y RA valen 1, el mensaje es una respuesta recursiva.
- **RCODE** es un código de respuesta de 4 bits. Los códigos posibles son: ningún error (0), error de formato (1), error interno del servidor (2), dominio inexistente (3), operación no implementada (4) o consulta rechazada (5).
- **QDCOUNT, ANCOUNT, NSCOUNT, ARCOUNT** son números de 16 bits que indican cuántas entradas hay en cada sección del mensaje: pregunta, respuesta, autoridad e información adicional, respectivamente.

Sección de pregunta

Los mensajes correspondientes a las consultas dispondrán de una o más entradas (por lo general, una) en la sección de pregunta. Cada entrada está formada por tres campos:

- **QNAME**: nombre de dominio del que el cliente desea obtener información.

ANOTACIONES

- **QTYPE**: número de 16 bits que indica el tipo de registro de recurso que el cliente quiere obtener como respuesta. El valor de dicho campo puede ser cualquiera de los números que pueden encontrarse en el campo TYPE de un registro, o uno de los siguientes:
  - **XFR** (código 252): se utiliza para solicitar la transferencia de todos los registros de una zona (cuando un servidor secundario quiere refrescar su base de datos).
  - **MAILB** (código 253): sirve para solicitar todos los registros relacionados con buzones (MB, MG y MR).
  - ***** (código 255): sirve para pedir todos los registros de cualquier tipo correspondientes a un nombre.
- **QCLASS:** número de 16 bits que indica la clase de registros deseados. Su valor puede ser el del campo CLASS de los registros o el valor especial "*" (código 255), que representa todas las clases.

Sección de respuesta

Los mensajes correspondientes a las respuestas contienen en la sección de respuesta el conjunto de registros de recurso que satisfacen los criterios dados por los campos de la consulta. Es decir, el servidor envía en la respuesta los registros correspondientes al nombre solicitado que concuerdan con el tipo o con los tipos requeridos y en la clase o clases requeridas.

Un caso especial es el de los alias. Si el nombre solicitado corresponde a un alias y el campo QTYPE es diferente de CNAME, el servidor debe repetir automáticamente la consulta sustituyendo el nombre original por el nombre canónico. De este modo, el cliente obtendrá directamente los datos deseados tanto si utiliza un alias, como si emplea el nombre canónico.

La respuesta a una petición AXFR consistirá en una secuencia de mensajes. El primero debe contener el registro SOA de la zona; en los siguientes deben encontrarse el resto de los registros, y el último debe ser otra vez el registro SOA, para que el receptor sepa que ya ha acabado la transferencia.

Sección de autoridad

En una respuesta, la sección de autoridad puede contener registros que referencien un servidor con autoridad para responder la consulta (para el caso de las respuestas en modo no recursivo).

Sección de información adicional

La sección de información adicional puede contener otros registros de recurso relacionados con la consulta, pero que no forman parte de la respuesta.

16.4.3. Representación de los registros de recurso

Cuando un mensaje debe contener una cadena de caracteres, se representa por medio de 1 byte, que indica su longitud y, a continuación, los caracteres de la cadena. Un nombre de dominio se representa como una concatenación de cadenas, una para cada una de las etiquetas que lo forman. Como los nombres de dominio acaban con la etiqueta del nodo raíz, una cadena que tenga por longitud 0 (y, por tanto, ningún carácter a continuación) indica el final del nombre. Para simplificar la implementación del servicio, la especificación RFC 1035 requiere que la longitud total de un nombre de dominio no sea superior a 255 bytes.

Nota

Representación comprimida de los nombres de dominio

La especificación RFC 1035 también define una representación comprimida de los nombres de dominio cuando hay muchos iguales, o que acaban igual, en un mensaje (como es el caso de los que contienen registros `SOA`). En lugar de una etiqueta, en un nombre puede haber 1 byte con los 2 bits de más peso a 1 (por tanto, no se puede confundir con la longitud de una etiqueta, que como máximo puede ser 63). Los otros 6 bits y los 8 del byte siguiente indican una posición dentro del mensaje en que se encuentra la continuación del nombre.

ANOTACIONES

La representación de los registros de recurso que pueden aparecer en las secciones de respuesta, autoridad e información adicional sigue el formato siguiente:

Figura 78.



Los campos que forman los registros de recurso dentro de los mensajes DNS son los siguientes:

- El **campo NAME** es el nombre de dominio correspondiente al registro.
- El **campo TYPE** es un número de 16 bits que indica el tipo de registro, según la tabla siguiente:

Tabla 7.

Número	Tipo de registro
1	A
2	NS
5	CNAME
6	SOA
7	MB
8	MG
9	MR
10	NULL
11	WKS
12	PTR
13	HINFO
14	MINFO
15	MX
16	TXT

ANOTACIONES

- El **campo `CLASS`** es un número de 16 bits que indica la clase del registro. Por norma general, su valor es 1, que corresponde a la clase Internet.

- El **campo `TTL`** es un número de 32 bits que indica el tiempo de vida del registro en segundos.

- El **campo `RDLENGTH`** es un número de 16 bits que indica cuántos bytes hay en el campo `RDATA`.

- El **campo `RDATA`** contiene los datos del registro, y su formato depende del valor del campo `TYPE`:

  – En los registros `A` (de la clase Internet), es un número de 32 bits.

  – En los registros `CNAME`, `NS`, `PTR`, `MB`, `MG` y `MR`, es un nombre de dominio.

  – En los registros `SOA`, es una secuencia de dos nombres de dominio (`MNAME` y `RNAME`) seguida de cinco números de 32 bits (`SERIAL`, `REFRESH`, `RETRY`, `EXPIRE` y `MINIMUM`).

  – En los registros `MX`, es un entero de 16 bits (la preferencia) seguido de un nombre de dominio.

  – En los registros `HINFO`, es la concatenación de dos cadenas de caracteres. La primera indica el tipo de UCP del ordenador, y la segunda, el sistema operativo.

  – En los registros `WKS`, es un número de 32 bits, como en los registros `A`, seguido de un número de 8 bits que identifica un protocolo (por ejemplo, 6 = TCP; 17 = UDP) y una secuencia de bits en la que cada bit indica si el ordenador ofrece un determinado servicio o no.

  – En los registros `MINFO`, es la concatenación de dos nombres de dominio. El primero constituye la dirección del responsable del buzón o lista de distribución, y el segundo, la dirección donde deben enviarse las notificaciones de error.

  – En los registros `TXT`, es una cadena de caracteres.

  – En los registros `NULL`, puede ser cualquier cosa.

16.5. Implementaciones del DNS

Prácticamente todos los ordenadores conectados a la red Internet incorporan alguna implementación de un cliente DNS (por norma general, una librería de funciones llamadas por las aplicaciones) para poder acceder a otros ordenadores conociendo los nombres de los mismos. Éste es el caso, por ejemplo, de los sistemas Unix o GNU/Linux, que proporcionan funciones como `gethostbyname` o `gethostbyaddr`.

En muchos sistemas Unix o GNU/Linux también se encuentra disponible una utilidad denominada `nslookup`, que sirve para efectuar consultas directamente a un servidor DNS. En el modo de funcionamiento básico, acepta comandos como los siguientes:

- *`nombre`*: envía una consulta utilizando el nombre especificado como valor del campo `QNAME` y muestra los resultados obtenidos, indicando si la respuesta es sin autoridad. En caso de serlo, informa de qué registros ha enviado el servidor en la sección de autoridad de la respuesta.

- `server` *`servidor`*: cambia el servidor por defecto al que se envían las consultas.

- `ls` *`dominio`*: muestra una lista de nodos del dominio e información asociada; si la lista es muy larga, se puede guardar en un fichero añadiendo `> fichero`.

- `set querytype=`*`tipo`*: permite cambiar el campo `QTYPE` de las consultas que, por defecto, es `A`, a cualquier otro valor (`NS`, `MX`, `PTR`, `CNAME`, etc.).

- `set q=`*`tipo`*: es equivalente a `set querytype`.

- `set domain=`*`origen`*: cambia el dominio origen que debe utilizarse cuando se especifica un nombre relativo.

- `set` *`opción`*: cambia el valor de alguna opción utilizada por el programa. Si el valor es `recurse`, pide que las consultas se lleven a cabo en modo recursivo. Si el valor es `norecurse`, pide que se rea-

ANOTACIONES

licen en modo no recursivo. Si el valor es `vc`, pide que se utilicen "circuitos virtuales", es decir, el TCP. Si el valor es `novc`, pide que se empleen datagramas, o lo que es lo mismo, el UDP, etc.

- `set all`: muestra el valor actual de los parámetros con que trabaja el programa (servidor por defecto, opciones, lista de nodos origen que debe utilizarse cuando se consultan nombres de dominio relativos, etc.).

- `?`: muestra un mensaje de ayuda.

- `exit`: acaba el programa.

Actividad

Utilizad el programa `nslookup` para responder a estas preguntas:

a) ¿Cuál es la dirección del ordenador `www.uoc.es`?
b) ¿Qué ordenador u ordenadores tienen por alias `ftp.uoc.es`?
c) ¿Cuál es el servidor DNS del dominio `uoc.es` ? ¿y el del dominio `uoc.edu`?
c) ¿Cuál es el servidor del dominio de correo `campus.uoc.es`?

ANOTACIONES

17. Servicios básicos de Internet

17.1. Terminal virtual: el protocolo Telnet

Uno de los servicios básicos para los que se puede utilizar una red de comunicaciones entre computadores es el acceso por medio de un terminal. Cuando se desarrollaron los sistemas multiusuario, la manera habitual de trabajar en los mismos consistía en utilizar un equipo terminal (en un inicio, un teletipo y, más adelante, un dispositivo con teclado y pantalla) conectado directamente al ordenador, en general por medio de una línea serie. El ordenador debía saber qué terminales tenía conectados y cómo podía controlarlos.

Nota

El hecho de que en una primera época se utilizaran teletipos es el motivo por el cual en UNIX se utilizó la abreviación `tty` (*teletype*) para designar los terminales (básicamente, impresoras con teclado).

Con la llegada de las redes de comunicaciones, que permitían interconectar diferentes ordenadores, uno de sus usos naturales era mantener una sesión de trabajo interactiva con un ordenador remoto sin necesidad de utilizar los terminales que estuvieran conectados al mismo directamente. Para ello, era preciso establecer un protocolo que, por un lado, hiciera posible que el ordenador remoto y el sistema en que quería trabajar el usuario se pudieran comunicar y se entendieran (es decir, un protocolo en el ámbito de aplicación, puesto que el problema del transporte se supone que ya estaba resuelto) y, por otro lado, facilitara el control del terminal del usuario desde el ordenador remoto.

Un ordenador puede permitir que se acceda al mismo desde cualquier terminal de cualquier otro ordenador conectado a la red; sin embargo, no es práctico que deba saber cómo se controlan todos los tipos posibles de terminal. La solución general a estos requisitos se basa en el uso de un protocolo de terminal virtual.



Un **terminal virtual** es un dispositivo imaginario para el que se definen unas funciones de control canónicas, de manera que se puede establecer una correspondencia entre ellas y las de cada tipo de terminal real.

ANOTACIONES

Lectura complementaria

Si queréis más información sobre el protocolo Telnet, consultad la obra siguiente:

J. Postel; J.K. Reynolds (1983, 1 de mayo). *RFC 854 - Telnet Protocol Specification.*

En el entorno Internet, el protocolo de terminal virtual más utilizado es **Telnet**, cuya especificación se publicó en 1983 en el estándar RFC 854.

17.1.1. Principios básicos del protocolo Telnet

El protocolo Telnet está basado en el protocolo de transporte TCP. En una comunicación Telnet, por norma general se sigue el modelo cliente/servidor; es decir, el sistema usuario establece una conexión con el sistema proveedor, que está esperando peticiones de conexión en un puerto determinado. Se puede utilizar cualquier número de puerto para las conexiones y, de hecho, existen muchas aplicaciones que utilizan el protocolo Telnet para la comunicación, cada una con su propio número. La aplicación básica, sin embargo, consiste en establecer una sesión de trabajo interactiva con el sistema servidor y, en este caso, el número de puerto utilizado es el 23.



Para resolver el problema del control del terminal en la aplicación de sesiones interactivas, en el protocolo Telnet se utiliza el concepto de **terminal virtual de red** o **NVT**. Un NVT es un terminal virtual con una funcionalidad muy básica, definida en la misma especificación del protocolo.

Nota

El usuario del terminal puede ser una persona o un proceso.

Los datos leídos en el primer caso por norma general serán los caracteres tecleados y, en el segundo, los que vaya generando el proceso. La presentación de los datos recibidos puede consistir en escribirlos en la pantalla, enviarlos a la impresora o pasarlos al proceso usuario.

Cuando se establece la conexión entre el cliente y el servidor, en un inicio se supone que la comunicación se produce entre dos NVT. Ello significa que tanto el sistema cliente como el sistema servidor deben mapear sus características en las de un NVT y suponer que en el otro extremo de la conexión hay otro NVT. En el modo de operación normal, cada terminal acepta datos del usuario y los envía por medio de la conexión establecida en el otro terminal, así como acepta los datos que llegan por la conexión y se los presenta al usuario.

Las características principales de los NVT son las siguientes:

a) Los datos se intercambian en bytes de 8 bits. Cuando representan caracteres, se utiliza la codificación ASCII (por tanto, se envía un código de 7 bits en cada byte).

ANOTACIONES

b) Existen tres caracteres de control que el NVT siempre los interpreta de la manera siguiente:

- `NUL` (carácter de control nulo, código 0): no es necesario hacer nada.
- `LF` (avance de línea, código 10): mover la posición de escritura actual (en una impresora, el carro, y en una pantalla, el cursor) una línea adelante manteniendo la posición horizontal.
- `CR` (retorno de carro, código 13): mover la posición de escritura atrás hasta el principio de la línea actual.

Por tanto, un final de línea (es decir, un salto al principio de la línea siguiente) se podría representar con las secuencias `CR-LF` o `LF-CR`. Sin embargo, para facilitar el mapeo en terminales que no tienen funciones separadas para el avance de línea y el retorno de carro, el protocolo Telnet especifica que los finales de línea deben representarse con la secuencia `CR-LF`, y los retornos de carro sin mover la posición vertical, con la secuencia `CR-NUL`. De este modo, cuando el terminal recibe un carácter `CR`, puede esperar a que llegue el siguiente, que deberá ser `LF` o `NUL`, para saber qué acción debe llevar a cabo.

c) Hay cinco caracteres más que opcionalmente pueden ser interpretados por el NVT y cuyo significado es el siguiente:

- `BEL` (timbre, código 7): generar una señal que, por lo general, será audible, pero que alternativamente puede ser visible, sin mover la posición actual.
- `BS` (retroceso, código 8): mover la posición horizontal un espacio atrás.
- `HT` (tabulación horizontal, código 9): mover la posición horizontal hacia delante hasta la posición de tabulación siguiente.
- `VT` (tabulación vertical, código 11): mover la posición vertical hacia delante hasta la posición de tabulación vertical siguiente.
- `FF` (avance de página, código 12): mover la posición vertical hasta el principio de la página siguiente, sin mover la posición horizontal.

ANOTACIONES

Cualquier otro carácter de control se considera equivalente a NUL.

d) El dispositivo NVT es *full duplex*; sin embargo, por norma general trabaja en modo *half duplex*; es decir, sólo envía los datos del usuario cuando ha leído una línea completa o cuando recibe alguna otra señal local que provoque la transmisión.

e) Si el usuario, de momento, no dispone de más datos para enviar, y se han procesado todos los datos recibidos, el terminal puede transmitir al sistema remoto una señal, denominada *Go Ahead* (GA), para indicarle que es su turno de enviar datos.

Nota

Sólo debería utilizarse la señal GA cuando el otro sistema no tuviera otra manera de saber que no debe esperar más datos. Una situación típica es que el usuario (cliente) genere líneas de una en una y el sistema remoto (servidor) envíe respuestas de cero, una o más líneas a cada una. En este caso, sólo será necesario que el servidor envíe la señal GA para notificar que el control pasa al usuario del terminal, y no será preciso que el cliente se lo envíe después de cada línea.

f) El NVT también puede generar, a instancias del usuario, dos señales más, denominadas *Break* (BRK) y *Synch*. El significado de la primera depende de la aplicación. La segunda sirve para cancelar los datos que haya enviado el terminal y todavía no hayan sido procesados por el sistema remoto.

Nota

Veremos el significado de las funciones de control en el apartado 18.1.2.

g) Asimismo, existen cinco funciones de control que puede generar el NVT a instancias del usuario: *Interrupt Process* (IP), *Abort Output* (AO), *Are You There* (AYT), *Erase Character* (EC) y *Erase Line* (EL).

Si el cliente y el servidor soportan una funcionalidad más avanzada que la de un NVT, que es lo más habitual, el protocolo permite que lleven a cabo una negociación para ponerse de acuerdo sobre qué características diferentes de las de un NVT utilizarán en la comunicación.

ANOTACIONES

La **negociación** consiste en intercambiar códigos que indican las opciones del protocolo que cada parte desea o está dispuesta a utilizar. Cuatro son los códigos que se utilizan para implementar la negociación:

- `DO` es una petición que se envía para pedir que se utilice una determinada opción.
- `WILL` es un ofrecimiento que se envía para indicar que el sistema está preparado para utilizar una determinada opción.
- `DON'T` significa que el sistema no quiere que se utilice la opción indicada.
- `WON'T` significa que el sistema no está preparado para utilizar la opción indicada.

Nota

El mecanismo de negociación es simétrico, de manera que cualquiera de las dos partes puede enviar un `DO` o un `WILL`. Tras enviar un `DO`, puede recibirse un `WILL` como respuesta positiva o un `WON'T` como respuesta negativa. Desde un punto de vista análogo, después de enviar un `WILL`, se puede recibir un `DO` como respuesta positiva o un `DON'T` como respuesta negativa. Ello significa que un `DO` puede actuar como petición o como confirmación si la otra parte ha enviado un `WILL`, y viceversa.

Por norma general, la negociación de las opciones se produce al inicio de la conexión, pero también es posible efectuarla en cualquier otro momento. Cada sistema debe empezar a utilizar una opción cuando envía o recibe la respuesta positiva correspondiente.

Este mecanismo de negociación está diseñado para que sea fácilmente extensible. Si un sistema no reconoce una determinada opción que se le pide o se le ofrece, simplemente debe contestar con `WON'T` o `DON'T`.

Nota

Las opciones Telnet posibles no estan definidas en la especificación del protocolo, sino en especificaciones independientes.

ANOTACIONES

Nota

Muchas opciones definen sus comandos para la subnegociación, como IS (para enviar el valor de un parámetro), SEND (para pedir que la otra parte lo envíe), etc.

En ocasiones se necesita más información para negociar una opción, como el valor de algún parámetro. En este caso, en primer lugar deben ponerse de acuerdo las dos partes con el mecanismo básico de los DO/ DON'T/WILL/WON'T para utilizar la opción deseada y, una vez se haya aceptado, debe llevarse a cabo una **subnegociación** de los valores de los parámetros utilizando una sintaxis específica de cada opción.

17.1.2. Comandos del protocolo Telnet

En los datos intercambiados entre dos sistemas que se comunican con el protocolo Telnet, se pueden intercalar ciertos comandos propios del protocolo, tales como la señal GA o los códigos de negociación. Para distinguir los comandos de los datos normales, los primeros deben ir prefijados con un código especial llamado IAC (Interpret As Command), que se representa con un byte igual a 255.

Por consiguiente, cada comando se representa con una secuencia de dos bytes, en la que el primero es el código IAC, y el segundo, el código propio del comando, excepto los comandos de negociación, que disponen de un tercer byte que sirve para indicar a qué opción se refieren. Para representar un byte normal de datos que sea igual a 255, es preciso prefijarlo con un código IAC. Cualquier otro byte de datos se representa directamente con su código.

Los comandos definidos en el protocolo Telnet son los siguientes:

- NOP (*No Operation*, código 241): operación nula.
- GA (*Go Ahead*, código 249): señal GA.
- BRK (*Break*, código 243): señal BRK.
- DO (código 253) + código opción: código de negociación DO.
- DON'T (código 254) + código opción: código de negociación DON'T.
- WILL (código 251) + código opción: código de negociación WILL.
- WON'T (código 252) + código opción: código de negociación WON'T.

ANOTACIONES

- `SB` (*Subnegotiation Begin*, código 250) + código opción: los datos que vengan a continuación corresponden a la subnegociación de una opción.
- `SE` (*Subnegotiation End*, código 240): indica el fin de los datos de subnegociación.
- `DM` (*Data Mark*, código 242): indica en qué punto de la secuencia de datos se ha enviado una señal *Synch*.
- `IP` (*Interrupt Process*, código 244): código de función que puede enviar el usuario para indicar al sistema remoto que interrumpa el proceso que está ejecutando.
- `AO` (*Abort Output*, código 245): código de función que sirve para indicar que se continúe ejecutando el proceso remoto, pero que deje de enviar al usuario los datos que genera.
- `AYT` (*Are You There*, código 246): código de función que pide al sistema que, cuando lo reciba, conteste con algún mensaje que sirva al usuario para verificar que la conexión continúa establecida (por ejemplo, cuando el proceso que ejecuta tarda en generar una respuesta).
- `EC` (*Erase Character*, código 247): código de función que sirve para borrar el último carácter escrito en el terminal.
- `EL` (*Erase Line*, código 248): código de función que sirve para borrar toda la línea actual.

El protocolo Telnet también proporciona un mecanismo para transmitir la señal *Synch*. Esta última no se representa con un comando como los otros, dado que debe tener un efecto inmediato. En este caso, se envían datos urgentes TCP acabados con un código DM y, en la secuencia de datos normales, se inserta otro código DM. Al recibir los datos urgentes, el sistema remoto pasará a procesar los datos normales en modo urgente (que puede consistir en descartar todo lo que haya en estos datos, salvo los comandos especiales) hasta que encuentre el DM.

El terminal puede generar automáticamente una señal *Synch* cuando el usuario envía alguna de las funciones que requieren atención inmediata, como `IP`, `AO` o `AYT` (pero no `EC` o `EL`).

ANOTACIONES

17.1.3. Implementaciones del protocolo Telnet

Hay implementaciones de servidores Telnet prácticamente en todos los sistemas multiusuario que utilizan el protocolo TCP. Los clientes Telnet son más numerosos todavía, porque también hay para sistemas monousuario.

Un ejemplo de implementación de cliente Telnet es la utilidad del sistema operativo GNU/Linux denominada precisamente `telnet`. Si se llama sin argumentos, entra en modo comando. Con un argumento, que puede ser una dirección IP o un nombre de servidor, establece una conexión con el puerto Telnet (el 23) de este servidor y entra en modo conexión. Con un segundo argumento, que puede ser un número de puerto o un nombre de servicio, establece la conexión con este puerto.

Cuando el número de puerto utilizado es el 23, el cliente inicia automáticamente el proceso de negociación enviando los códigos `DO` y `WILL` correspondientes a las opciones que soporta. Con cualquier otro puerto, por norma general no se envía ningún código de negociación, salvo que se reciba alguno del sistema remoto.

Cuando el programa está en modo conexión, envía al sistema remoto cada carácter que teclea el usuario. Desde el modo conexión se puede pasar al modo comando por medio del carácter de escape, que suele ser "`^]`". En este modo, el programa admite, entre otros, los comandos siguientes:

Nota

Las nombres de los comandos se pueden abreviar siempre que la abreviatura no genere ambigüedades.

- `open`: establece una conexión con el servidor, y opcionalmente con el puerto, indicado en los argumentos de este comando.
- comando nulo (línea vacía): si hay una conexión establecida, sale del modo comando y vuelve al modo conexión.
- `send`: envía al sistema remoto el código Telnet indicado por el argumento, que puede ser `ao`, `ayt`, `brk`, `ec`, `el`, `ga`, `ip`, `synch`, etc.
- "`^]`" (o el carácter que actúe como carácter de escape): envía este último al sistema remoto (equivale a `send escape`).

ANOTACIONES

- `?`: muestra un mensaje de ayuda.

- `quit`: cierra la conexión, en caso de haber conexión establecida, y acaba el programa (que también acaba cuando el sistema remoto cierra la conexión).

Actividad

Con algunas implementaciones Unix del cliente Telnet, se puede utilizar el comando `toggle netdata` para ver el diálogo de la negociación. Probad esta otra opción en una sesión interactiva (en el puerto por defecto 23) y en una conexión a otro puerto (por ejemplo, `echo`).

17.2. Terminal virtual en GNU/Linux: el protocolo rlogin

Cuando se desea establecer una sesión de trabajo interactiva desde un sistema Unix con otro sistema Unix, además de utilizar el protocolo Telnet, también existe la posibilidad de utilizar el protocolo llamado **rlogin**. Este último se desarrolló en un inicio en el entorno de las utilidades de comunicaciones de la variante de Unix llamada BSD; sin embargo, hoy día hay implementaciones disponibles en todas las variantes, así como en otros sistemas operativos. La especificación del protocolo rlogin está publicada en el documento *RFC 1282*.

Lectura complementaria

Para obtener más información sobre el protocolo rlogin, consultad la obra siguiente:
B. Kantor (1991, diciembre). *RFC 1282 - BSD Rlogin*.

Las características principales que diferencian el protocolo rlogin del protocolo Telnet son las siguientes:

a) El servidor siempre es informado del tipo de terminal con que trabaja el cliente sin necesidad de negociación.

b) El servidor también es informado de la identidad del usuario en el sistema cliente, lo que se puede utilizar para automatizar el proceso de autenticación.

c) Si cambian los tamaños de la ventana de presentación del terminal, se puede enviar automáticamente una señal al servidor para notificárselo.

ANOTACIONES

d) En la transmisión de datos, siempre pueden utilizarse los 8 bits de cada byte.

17.2.1. Conceptos básicos del protocolo rlogin

El protocolo rlogin utiliza conexiones TCP. El nombre oficial del servicio proporcionado por este protocolo es `login` y el número de puerto asignado a este servicio es el 513.

Cuando el cliente establece la conexión con el servidor, en primer lugar le envía cuatro cadenas de caracteres, cada una acabada con un carácter `NUL` (código 0). Son las cadenas siguientes:

- Una cadena vacía (sólo contiene el carácter `NUL`).
- El nombre del usuario en el sistema cliente.
- El nombre de usuario con que se quiere establecer la sesión de trabajo en el sistema servidor.
- El tipo de terminal y, separado con "/", su velocidad (por ejemplo, vt100/9600).

Cuando ha recibido estas cuatro cadenas, el servidor envía un carácter `NUL` y empieza la transferencia de datos entre cliente y servidor en modo control de flujo. En este modo, el cliente envía los caracteres que teclea el usuario tal como le llegan, excepto los caracteres de control DC3 ("`^S`") y DC1 ("`^Q`"), que significan 'suspender la transmisión' y 'retomarla', respectivamente.

Por otro lado, el cliente presenta los datos que envía el servidor tal como le llegan. El cliente también puede recibir mensajes de control del servidor, que se representan con un código de un byte enviado en datos urgentes TCP. El cliente debe responder a estos mensajes de manera inmediata y suspender temporalmente el procesado de otros datos que pueda haber recibido.

17.2.2. Implementación del protocolo rlogin

En los sistemas GNU/Linux hay una utilidad llamada `rlogin` que actúa como cliente del protocolo rlogin. Es preciso que se le especifique

el nombre del servidor remoto y, de manera optativa, el nombre de usuario que debe utilizar en este servidor, con la opción `-l` (por defecto, se utiliza el mismo nombre de usuario que en el sistema local).

La mayoría de los servidores rlogin implementan la autenticación automática de la manera siguiente:

- Si el usuario remoto tiene en su directorio un fichero llamado `.rhosts` con una línea en la que se encuentre el nombre del ordenador cliente y, opcionalmente, el nombre del usuario cliente (en ausencia de este nombre de usuario, se toma el del usuario en el servidor), se inicia directamente la sesión interactiva sin necesidad de pedir ninguna contraseña.
- Si no dispone de este fichero, se consulta otro, el `/etc/hosts.equiv`, para ver si existe alguna línea con el nombre del ordenador cliente (en este caso, debe coincidir el nombre del usuario en el cliente y en el servidor) y, si está, también se da al usuario por autenticado.
- En cualquier otro caso, se sigue el procedimiento normal de autenticación mediante contraseña.

Nota

Por motivos de seguridad, los servidores rlogin suelen requerir que el fichero `.rhosts` tenga permiso de lectura sólo para el usuario a quien pertenece. Por otro lado, los nombres de servidores en los ficheros `.rhosts` y `/etc/hosts.equiv` no pueden ser alias, deben ser los nombres oficiales. Y, como precaución adicional, el fichero `/etc/hosts.equiv` sólo se utiliza cuando el nombre del usuario en el servidor es diferente de `root`.

Antes de consultar los ficheros `.rhosts` y `/etc/hosts.equiv`, el servidor siempre comprueba que el cliente se haya conectado desde un puerto reservado (con número inferior a 1024). En caso contrario, no se permite la autenticación automática, puesto que la conexión podría provenir de otro usuario que estuviera ejecutando un programa que siguiese el protocolo rlogin, pero que enviase información falsa sobre su identidad.

ANOTACIONES

La autenticación automática puede resultar cómoda porque ahorra teclear las contraseñas, pero también puede ser una fuente de problemas si no se utiliza con precaución.

Además de leer los caracteres tecleados y enviárselos al servidor, el cliente también puede recibir comandos del usuario. Para distinguirlos de los caracteres normales, se representan con un carácter precedido del símbolo "~". Esta secuencia sólo se reconoce al inicio de las líneas (es decir, inmediatamente a continuación de un salto de línea). El carácter del comando puede ser, por ejemplo, el siguiente:

- "`^Z`" (o el carácter correspondiente a la función SUSP del terminal): suspende la ejecución del programa cliente, con la posibilidad de retomarla más adelante (la manera de hacerlo puede depender del *shell*, pero por lo general es con el comando `fg`).

- "`^Y`" (o el carácter correspondiente a la función DSUSP del terminal): suspende el envío de datos al servidor y, por consiguiente, el teclado queda disponible para otros procesos; sin embargo, la recepción continúa (si se reciben nuevos datos, se mostrarán cuando lleguen).

- "~": envía el carácter ~.

- "`.`": cierra la conexión.

Si el carácter no corresponde a ningún comando reconocido por el cliente, se envía al servidor tal como ha llegado, precedido por el carácter "~".

17.3. Otros servicios

17.3.1. Ejecución remota con autenticación automática: rsh

Paralelamente al servicio de terminal virtual, que permite mantener una sesión de trabajo en un sistema remoto, en los sistemas Unix

ANOTACIONES

BSD se desarrolló otro servicio que permite ejecutar un comando en el sistema remoto. El programa que actúa como cliente de este servicio se llama **rsh** (*remote shell*).

El nombre oficial de este servicio es `shell`, el protocolo de transporte utilizado es el TCP y el número de puerto asignado es el 514.

A continuación presentamos los pasos que se siguen en este protocolo:

1) El servidor comprueba que el cliente se haya conectado desde un puerto reservado. En caso contrario, cierra la conexión.

2) El cliente envía al servidor una cadena acabada en `NUL` que contiene un número decimal:

- Si es diferente de cero, se interpreta como un número de puerto y el servidor establece en este puerto una conexión secundaria (esta segunda conexión se establece también desde un puerto reservado).

- Si es cero, la misma conexión principal actúa también como secundaria.

3) El cliente envía tres cadenas más acabadas en `NUL` con el nombre del usuario en el sistema servidor, el nombre en el sistema cliente y el comando a ejecutar.

4) El servidor comprueba en los ficheros `.rhosts` y/o `/etc/hosts.equiv` que el usuario esté autorizado para la autenticación automática.

5) Si lo está, el servidor envía un carácter `NUL` por la conexión secundaria; en caso contrario, cierra las conexiones.

6) El servidor ejecuta el comando indicado con la identidad del usuario indicado. Los datos que escriba este comando por la salida estándar (`stdout` en la nomenclatura del lenguaje C) se enviarán por la conexión principal, y los que escriba por la salida de error (`stderr`), por la conexión secundaria. Las que envíe el cliente serán accesibles por la entrada estándar (`stdin`).

7) Cuando acabe la ejecución del comando, el servidor cerrará la conexión.

ANOTACIONES

17.3.2. Ejecución remota: rexec

Con rsh, el usuario debe estar autorizado por medio de los ficheros `.rhosts` o `/etc/hosts.equiv` del sistema remoto para poder ejecutar un comando en el mismo. Existe otro servicio muy parecido en que no se utiliza la autenticación automática, sino que siempre requiere que el usuario proporcione su contraseña en el sistema remoto. En algunos sistemas Unix, se encuentra disponible un programa llamado ***rexec*** que actúa como cliente de este servicio. En otros, sólo hay una función de la librería estándar, *rexec*, que implementa el protocolo.

El nombre oficial de este servicio es `exec`, el protocolo de transporte utilizado es el TCP y el número de puerto asignado es el 512.

Este protocolo es muy parecido al anterior: las únicas diferencias son que, en lugar de enviar el nombre del usuario, en el sistema cliente se envía la contraseña del usuario en el sistema servidor (los ficheros `.rhosts` y `/etc/hosts.equiv` no se consultan) y que no es preciso verificar que las conexiones provengan de puertos reservados.

17.3.3. Servicios triviales

La mayoría de los sistemas Unix y GNU/Linux proporcionan una serie de servicios denominados *triviales*, que se pueden utilizar para llevar a cabo pruebas de funcionamiento de los módulos que implementan los protocolos de transporte. Todos son accesibles por medio de TCP y UDP, y el número de puerto utilizado en ambos casos es el mismo. Algunos ejemplos de servicios triviales son los siguientes:

- `echo` (puerto 7): retorna todos los bytes (en TCP) o datagramas (en UDP) que recibe (RFC 862).
- `discard` (puerto 9): lee datos (bytes o datagramas), pero no hace nada más (RFC 863).
- `chargen` (puerto 19): en TCP, cuando se establece la conexión, el servidor empieza a enviar una secuencia de caracteres hasta que el cliente la cierra y, en UDP, cuando el servidor recibe un datagrama, responde con otro que contiene un nombre arbitrario de caracteres (RFC 864).

ANOTACIONES

- `daytime` (puerto 13): cuando se establece la conexión, o se recibe un datagrama, el servidor envía la fecha y la hora actual en formato legible para los humanos (RFC 867).

- `time` (puerto 37): cuando se establece la conexión, o se recibe un datagrama, el servidor envía un número de 4 bytes que representa el número de segundos transcurridos desde el 1 de enero de 1970 a las 0 horas GMT (RFC 868).

Finger

`Name/Finger` es otro ejemplo de servicio sencillo que permite obtener información de un usuario en un sistema remoto. La especificación del protocolo para este servicio está recogida en el documento *RFC 742*.

El nombre oficial de este servicio es `finger`, el protocolo de transporte utilizado es el TCP y el número de puerto asignado es el 79.

El programa cliente se llama *finger* y, por norma general, admite argumentos de la forma `usuario`, `usuario@host` o `@host`:

- En el primer caso, proporciona información sobre un usuario en el sistema local sin necesidad de utilizar ningún protocolo.
- En el segundo, informa sobre un usuario de un sistema remoto.
- En el tercero, normalmente ofrece una información resumida de los usuarios que están actualmente conectados con sesiones interactivas al sistema remoto.

Cuando se establece la conexión, el cliente simplemente envía una línea al servidor acabada con `CR-LF`. En esta última se encuentra el nombre del usuario de quien se quiere obtener información. Si no hay ningún nombre, se entiende que la solicitud se refiere a todos los usuarios que estén conectados.

El formato de la respuesta depende de la implementación del servidor y puede incluir información como el nombre del usuario, su nombre real, cuándo se conectó por última vez, cuándo recibió o

ANOTACIONES

leyó el correo por última vez, etc., y, si está conectado actualmente, desde dónde, en qué terminal, cuánto tiempo hace que no hay actividad en su terminal, etc. La información retornada normalmente es un subconjunto de todos estos campos; sin embargo, si la línea de petición empieza con los caracteres "/W", el servidor puede retornar información más completa.

ANOTACIONES

18. Transferencia de ficheros

18.1. FTP: protocolo de transferencia de ficheros

Una de las primeras aplicaciones básicas desarrolladas en el entorno de lo que después sería la red Internet fue la transferencia de ficheros entre diferentes sistemas. Al principio de los años setenta ya se elaboraron las primeras especificaciones del protocolo más utilizado para esta finalidad, el **FTP**. Después de algunas modificaciones y mejoras, la especificación oficial del protocolo se publicó en 1985 en el documento *RFC 959*.

El FTP se basa en el modelo cliente/servidor y permite la transferencia de ficheros tanto del servidor al cliente, como del cliente al servidor. Asimismo, permite que un cliente efectúe transferencias directas de un servidor a otro, con lo que se ahorra la necesidad de copiar los ficheros del primer servidor al cliente y pasarlos después del cliente al segundo servidor.

El protocolo proporciona también operaciones para que el cliente pueda manipular el sistema de ficheros del servidor: borrar ficheros o cambiarles el nombre, crear y borrar directorios, listar sus contenidos, etc.

Uno de los objetivos principales de este protocolo consiste en permitir la interoperabilidad entre sistemas muy distintos, escondiendo los detalles de la estructura interna de los sistemas de ficheros locales y de la organización de los contenidos de los ficheros.

Lectura complementaria

Para más información sobre el FTP, consultad la obra siguiente:

J. Postel; J. Reynolds (1985, octubre). *RFC 959 - File Transfer Protocol*.

Nota

Algunos de los sistemas utilizados en la época de desarrollo del FTP actualmente están obsoletos; por

ANOTACIONES

ejemplo, los que utilizan palabras de 36 bits o códigos de caracteres incompatibles con el estándar ASCII o los que almacenan los ficheros en "páginas" no necesariamente contiguas. De hecho, muchas de las implementaciones actuales del protocolo no soportan estas características especiales previstas en la especificación.

18.1.1. El modelo del FTP

En el modelo general descrito en la especificación del FTP, tanto en el servidor como en el cliente hay dos entidades que intervienen en la transferencia de ficheros:

a) El **intérprete de protocolo** se encarga del intercambio de los comandos del protocolo. En la parte del cliente, las operaciones que el usuario solicita por medio de la interfaz de usuario, el intérprete las convierte en una secuencia adecuada de comandos FTP y se envían al servidor.

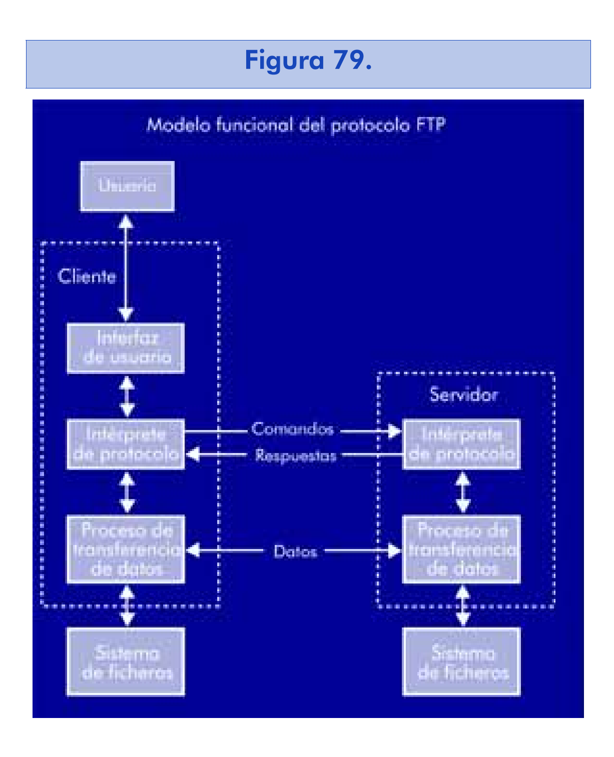
En la parte del servidor se interpretan los comandos recibidos, se generan las respuestas correspondientes y se envían al cliente. Por tanto, el intérprete cliente debe analizar estas respuestas, por ejemplo, para informar al usuario del resultado de la operación o para proseguir la secuencia de comandos FTP.

b) El **proceso de transferencia de datos**, que está bajo el control del intérprete de protocolo, se encarga de intercambiar los datos que deben transferirse, es decir, los contenidos de los ficheros o bien los listados de los directorios.

Tanto en la parte del cliente, como en la del servidor este proceso interactúa directamente con el sistema de ficheros locales para leer sus datos o para almacenarlos en los mismos.

ANOTACIONES

Figura 79.



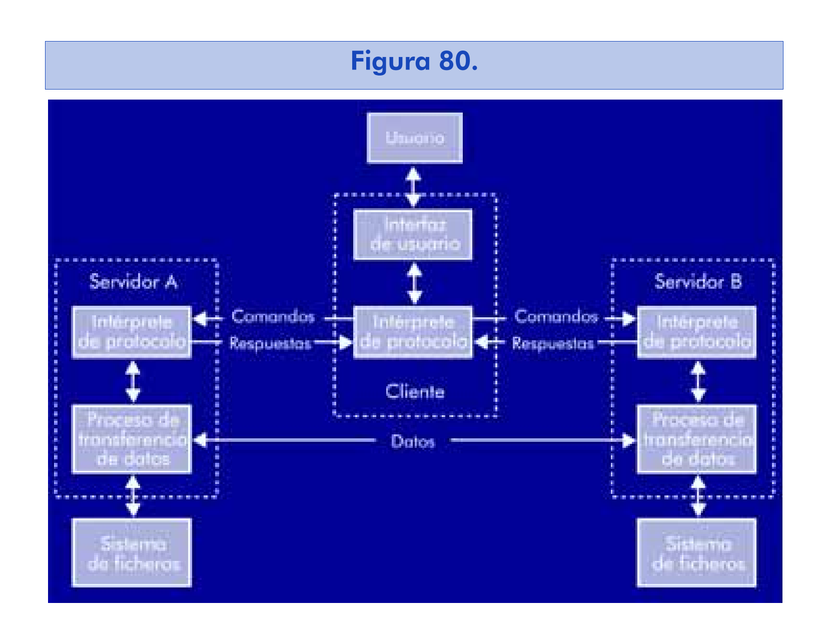
Los dos intérpretes de protocolo se comunican mediante una conexión TCP llamada *conexión de control*. Cuando deben transferirse datos de un sistema al otro, se establece otra conexión TCP entre los dos procesos de transferencia de datos denominada *conexión de datos*. Generalmente, la parte activa de esta conexión (la que la inicia) constituye el proceso de transferencia del servidor, y la parte pasiva, el del cliente.

Nota

El FTP permite efectuar más de una transferencia en una única sesión. Según el modo de transmisión, es posible que se abran y se cierren diferentes conexiones de datos durante una misma conexión de control, o bien que se utilice la misma conexión de datos para las diferentes transferencias.

La figura siguiente muestra la variación del modelo general para el caso en que un cliente controle una transferencia entre dos servidores (uno de los cuales actuará como servidor activo y el otro, como servidor pasivo):

Figura 80.



ANOTACIONES

18.1.2. Conceptos básicos del FTP

El FTP está basado en conexiones TCP. El intérprete de protocolo del servidor debe estar preparado para recibir peticiones de conexión en un puerto TCP que, por defecto, es el asignado oficialmente al servicio FTP: el número 21.

El intérprete de protocolo del cliente establece una conexión de control con el puerto del intérprete servidor. En esta conexión se utilizan las reglas especificadas en el protocolo Telnet. Ello significa que, por defecto, los intérpretes de protocolo se intercambian mensajes codificados con bytes de 8 bits, según el juego de caracteres ASCII, y representan los finales de línea con la secuencia `<CRLF>`.

Nota

Consultad la descripción de las reglas especificadas por el protocolo Telnet (RFC 854) en el apartado 18.1 de la unidad anterior.

Los **comandos FTP** constituyen los mensajes que envía el intérprete cliente, y los que envía el intérprete servidor son respuestas a dichos comandos. Las respuestas se generan siguiendo el orden en que el cliente envía los comandos, puesto que en general el servidor efectúa las operaciones de manera secuencial (no empieza una nueva operación hasta que no ha acabado la anterior). A continuación, analizamos por separado los comandos y las respuestas:

Nota

Las excepciones a la regla de la respuesta secuencial son los comandos obtener el estado actual de una transferencia, abortar una operación y cerrar la sesión.

1) **Comandos FTP**: un comando FTP se representa por medio de un código de comando de hasta cuatro letras (que pueden ser indistintamente mayúsculas o minúsculas), seguido de una lista de cero o más argumentos, separados por espacios, acabada con un final de línea.

La especificación RFC 959 define treinta y tres comandos agrupados en tres categorías diferentes (representamos entre paréntesis el código correspondiente a cada comando):

Nota

Algunas implementaciones de servidores FTP soportan otros códigos de comandos no estándares, que pueden tener más de cuatro letras.

a) **Comandos de control de acceso**: nombre de usuario (`USER`), contraseña (`PASS`), cuenta (`ACCT`), cambiar de directorio (`CWD`), cambiar al directorio padre (`CDUP`), montar un sistema de ficheros (`SMNT`), reinicializar (`REIN`), acabar la sesión (`QUIT`).

b) **Comandos de parámetros de transferencia**: estructura de fichero (`STRU`), modo de transmisión (`MODE`), tipo de representación (`TYPE`), puerto de datos (`PORT`), puerto pasivo (`PASV`).

ANOTACIONES

c) **Comandos de servicio FTP**: obtener (RETR), almacenar (STOR), almacenar con nombre único (STOU), añadir (APPE), listar (LIST), listar nombres (NLST), mostrar el directorio actual (PWD), reservar espacio (ALLO), abortar (ABOR), retomar (REST), borrar (DELE), nombre antiguo (RNFR), nombre nuevo (RNTO), crear un directorio (MKD), borrar un directorio (RMD), estatus (STAT), sistema (SYST), servicios adicionales (SITE), ayuda (HELP) y operación nula (NOOP).

2) **Respuestas FTP**: las respuestas constan de un código numérico de tres dígitos decimales, que permite al intérprete del protocolo cliente saber cuál es el resultado de la operación, seguido de un texto explicativo destinado al usuario humano.

El texto explicativo del mensaje de respuesta puede tener una línea o más:

- Si tiene una línea, la respuesta se representa con el código numérico seguido del carácter espacio y, a continuación, el texto y un final de línea.

- Si tiene más de una línea, la primera se representa con el código numérico seguido del carácter "–" y, a continuación, las líneas de texto, acabadas con finales de línea, la última de las cuales debe empezar con el mismo código numérico de la primera línea seguido del carácter espacio.

Ejemplo

Ejemplo de respuesta de una sola línea:

```
220 Sistema preparado. Introducid nombre de usuario y contraseña.
```

Ejemplo de respuesta con múltiples líneas:

```
220-Sistema preparado.
   Introducid el nombre de usuario y la
   contraseña para acceder a los ficheros
   restringidos, o utilizad el nombre
   "anonymous"
220 para acceder al directorio público.
```

Nota

De este grupo de comandos de servicio, los seis siguientes son los comandos de transferencia; es decir, los que utilizan la conexión de datos.

- RETR
- STOR
- STOU
- APPE
- LIST
- NLST

ANOTACIONES

Nota

La estructura de los códigos de respuesta y el significado de cada dígito son comunes a todos los protocolos. Podéis encontrar una explicación de los mismos en el anexo 4.

La especificación RFC 959 define los códigos de respuesta posibles para cada uno de los comandos y su significado.

Tabla 8.

Códigos de respuesta	
Código	**Significado**
110	Marca de reanudación.
120	Esperar hasta que el servidor esté preparado.
125	El servidor transfiere datos (la conexión de datos ya está abierta).
150	El servidor establece la conexión de datos e inicia la transferencia.
200	Operación efectuada.
202	Comando innecesario en este servidor.
211	Información de estado del sistema o mensaje de ayuda del sistema.
212	Información de estado del directorio.
213	Información de estado del fichero.
214	Mensaje de ayuda (destinado al usuario humano).
215	Tipo de sistema.
220	Servidor preparado.
221	El servidor cierra la conexión de control.
226	Operación efectuada y conexión de datos cerrada.
227	El servidor está en modo pasivo (`h1,h2,h3,h4,p1,p2`).
230	Proceso de autenticación completado satisfactoriamente.
250	Operación de fichero efectuada.
257	El directorio es el resultado de la operación.
331	Enviar contraseña.
332	Enviar cuenta.
350	Enviar el comando siguiente para completar la operación.
421	Servicio no disponible (por ejemplo, por *time out*...), conexión de control cerrada.
425	No se puede establecer la conexión de datos.
426	Transferencia abortada y conexión de datos cerrada.
450	No se puede efectuar la operación sobre el fichero (por ejemplo, fichero ocupado).
452	Espacio de disco insuficiente: transferencia no iniciada.
500	Error de sintaxis en el comando.

ANOTACIONES

Códigos de respuesta	
Código	**Significado**
501	Error en los argumentos.
502	Comando no implementado.
503	Error en la secuencia de comandos (este comando no corresponde).
504	Argumento no soportado.
530	El usuario no se ha autenticado correctamente.
532	Se necesita cuenta para esta operación.
550	No se puede acceder al fichero o directorio (no existe, acceso denegado, etc.).
552	Espacio de disco insuficiente: transferencia abortada.
553	Nombre de fichero no permitido.

Cuando ha establecido la conexión de control con el intérprete de protocolo del servidor y antes de enviar ningún comando, el cliente debe esperar a recibir una respuesta, que puede tener los códigos 220, 421 ó 120. Si el código es 120, el cliente debe esperar hasta que el servidor envíe una respuesta 220. Cuando el servidor indique que está preparado, puede empezar el intercambio de comandos y respuestas. Por norma general, el servidor cerrará la conexión de control cuando se lo solicite el cliente (con el comando de acabar sesión).

Si el cliente desea realizar **operaciones de transferencia**, su proceso de transferencia de datos debería estar preparado para recibir peticiones de conexión en un puerto TCP, que por defecto es el mismo desde el que el intérprete ha iniciado la conexión de control.

El encargado de establecer las conexiones de datos es el proceso de transferencia de datos del servidor, desde su puerto de datos. Por defecto, este puerto es $L - 1$ (donde L es el número de puerto correspondiente a la conexión de control). Es decir, si el servidor ha recibido la conexión de control en el puerto 21, su puerto de datos por defecto será el 20. Evidentemente, el cliente puede solicitar el uso de puertos diferentes de los predeterminados por medio de los comandos adecuados.

Generalmente, las conexiones de datos las cierra el servidor, excepto cuando sea la otra parte (el cliente o un servidor pasivo) quien envíe

Nota

El texto explicativo de las respuestas es de formato libre, excepto en el caso de las respuestas 110 (comando `RETR`), 227 (comando `PASV`) y 257 (comandos `PWD` y `MKD`).

ANOTACIONES

los datos y el modo de transmisión utilizado requiera el cierre de la conexión para indicar el final del fichero.

18.1.3. Funcionalidad del FTP

Las acciones que lleva a cabo el servidor para cada uno de los comandos definidos en la especificación RFC 959 son los siguientes:

1) Nombre de usuario (**USER**)

El argumento de este comando es el nombre que es preciso suministrar al sistema servidor para identificar al usuario con el objetivo de acceder a los ficheros. Éste suele ser el primer comando que envía el cliente. Asimismo, es posible enviarlo en medio de una sesión; entonces el servidor se olvida de la identidad del usuario anterior y realiza las nuevas operaciones bajo el nombre del nuevo usuario.

Si el servidor envía la respuesta 230, significa que el proceso de autenticación ya se ha completado; si envía la 530, significa que este usuario no es admisible (por ejemplo, porque no hay ningún ususario con este nombre) y, si envía la 331 o la 332, entonces se necesita una contraseña o una cuenta, respectivamente.

Nota

En muchos sistemas, un intento de conexión con un nombre de usuario inexistente dará como resultado una respuesta 331, para no proporcionar información a los usuarios ajenos al sistema sobre qué nombres son válidos y cuáles no.

El código 332 indica que la operación solicitada se llevará a cabo tan pronto como se reciba el comando ACCT.

2) Contraseña (**PASS**)

El argumento de este comando es la contraseña que necesita el sistema servidor para verificar la identidad del usuario. Si no se necesita ninguna, la respuesta será 202; si se necesita pero es incorrecta,

ANOTACIONES

530, y si es correcta, puede ser 230 (autenticación completada) o 332 (se necesita una cuenta).

3) Cuenta (**ACCT**)

Algunos sistemas pueden requerir que el usuario proporcione una identificación de cuenta, bien en el proceso de autenticación inicial, bien para efectuar determinadas operaciones, como almacenar ficheros. El servidor hará saber que necesita esta información enviando una respuesta 332 en el proceso de autenticación, o una respuesta 332 o 532 después de una determinada operación.

4) Estructura de fichero (**STRU**)

Este comando sirve para especificar cómo estarán estructurados los ficheros que deben transferirse. El tipo de estructura afecta al modo de transmisión, como veremos a continuación. Los valores posibles del argumento son los tres siguientes:

- `R`: el fichero consta de una secuencia de registros.
- `P`: la estructura se basa en páginas indexadas. (La opción P era útil en los sistemas de la época en que se desarrolló el FTP; hoy día está prácticamente en desuso.)
- `F`: el fichero se considera simplemente una secuencia de bytes (en este caso se considera que sólo hay estructura de fichero).

Si no se utiliza el comando `STRU`, el tipo de estructura por defecto es `F`.

5) Modo de transmisión (**MODE**)

El argumento indica cómo se transmitirán los ficheros. Puede tener los valores siguientes:

- `B`: la transmisión se lleva a cabo en bloques de datos, cada uno precedido de una cabecera con la longitud del bloque (2 bytes) y un código descriptor (1 byte). Este último sirve para indicar, por ejemplo, si el bloque es el último de un registro o del fichero.

ANOTACIONES

Nota

La combinación modo *stream* y estructura de fichero es la única que requiere cerrar la conexión de datos después de cada transferencia. En los otros casos, el servidor puede optar entre mantenerla abierta o cerrarla, e informa de la misma al cliente con una respuesta 250 o 226, respectivamente.

- `C`: la transmisión se realiza en bloques más compactos, con cabeceras de un solo byte, y permiten codificar una secuencia de hasta 64 bytes repetidos en un bloque de 2 bytes.

- `S`: la transmisión se efectúa en modo *stream*; es decir, como una simple secuencia de bytes. Si se utiliza con el tipo de estructura en registros, se insertan códigos de control en la secuencia para señalar los finales de registro y de fichero. Si el tipo de estructura es de fichero y la transmisión es en modo *stream*, la única manera de indicar el final de fichero es cerrando la conexión de datos.

Si no se utiliza el comando `MODE`, el modo de transmisión por defecto es `S`.

6) Tipo de representación (**`TYPE`**)

Con este comando, se especifica cómo se representarán los datos de los ficheros que deben transferirse. El proceso que los envía debe convertir el contenido de los ficheros en la representación especificada, y el proceso que los recibe, debe convertirlos en su representación local. Los valores posibles del argumento son los siguientes:

- `A`: se utiliza la especificación NVT definida en el protocolo Telnet (caracteres de 8 bits codificados en ASCII y finales de línea representados con la secuencia `<CRLF>`).

- `E`: se utilizan caracteres de 8 bits codificados en `EBCDIC`.

Nota

En los tipos de representación `A` y `E`, un segundo argumento opcional permite indicar cómo se especifica la información de control para el formato vertical del texto (separación entre párrafos, cambios de página, etc.) y puede valer `N` (no hay información de formato), `T` (hay caracteres de control Telnet: ASCII\EBCDIC) o `C` (se utiliza el formato del estándar ASA, RFC 740).

ANOTACIONES

- `I`: se envía una imagen del fichero consistente en una secuencia arbitraria de bits, codificados en bytes (8 bits), que el receptor debe almacenar tal como le llega (o añadiendo ceros al final si el sistema local necesita que la longitud total sea múltiplo de alguna cantidad).

- `L`: se envía una secuencia de bytes lógicos de *n* bits, siendo *n* el valor del segundo argumento (obligatorio en este caso), con todos los bits consecutivos agrupados en bytes de 8 bits. Según el valor de *n*, es posible que el receptor necesite aplicar alguna transformación (reversible) para almacenar los datos.

Nota

Sea cual sea el tipo de representación, la transferencia siempre se realiza en bytes de 8 bits.

Si no se utiliza el comando `TYPE`, el tipo de representación por defecto es `A` (y sin información de formato vertical).

7) Cambiar de directorio (`CWD`)

Normalmente, cada usuario tiene asignado un directorio por defecto en el sistema de ficheros del servidor. Este comando sirve para cambiar el directorio de trabajo por el directorio que indica el argumento.

8) Cambiar al directorio padre (`CDUP`)

Este comando no recibe ningún argumento. Se trata de un caso particular del anterior que el protocolo proporciona para facilitar el acceso al directorio padre del actual con independencia de la sintaxis utilizada en el sistema de ficheros para referenciarlo.

9) Montar un sistema de ficheros (`SMNT`)

El argumento es un nombre que, en la sintaxis del sistema de ficheros del servidor, permite seleccionar un nuevo grupo de ficheros (por ejemplo, otro sistema de ficheros, otra partición del disco, otro disco, etc.).

10) Mostrar el directorio actual (`PWD`)

En la respuesta a este comando (código 257), el servidor informa del directorio actual. Para facilitar la interpretación de la respues-

ta, el nombre del directorio debe ser la primera palabra que venga después del código numérico, y debe ir delimitado por el carácter “"”.

11) Puerto de datos (`PORT`)

Con este comando, el cliente indica cuál es su puerto de datos, en caso de que sea diferente del puerto por defecto. Para efectuar las transferencias, el servidor deberá establecer la conexión de datos en el puerto especificado. Si ya había una conexión de datos establecida con otro puerto cuando se recibe este comando, el servidor deberá cerrarla.

Nota

Recordad que el puerto de datos por defecto del cliente es el mismo desde el que ha establecido la conexión de control.

El argumento debe tener esta forma: *`h1,h2,h3,h4,p1,p2`* (cada uno de los elementos es un número decimal entre 0 y 255). Los cuatro primeros números forman la dirección IP, y los dos últimos, el número de puerto (que se representa de más a menos peso).

Nota

Cambio de puerto

Cuando el modo de transmisión requiere cerrar la conexión después de cada transferencia, se suele utilizar este comando para ir variando el número de puerto y evitar así las demoras que puede haber cuando se intenta reutilizar un puerto TCP que se acaba de cerrar.

El cliente puede especificar una dirección IP diferente de la suya; de esta manera, se puede efectuar una transferencia entre dos servidores, como ilustra la figura del modelo FTP, en que un cliente controla la transferencia de datos entre dos servidores.

Nota

Consultad la figura mencionada en el apartado 18.1.1.

12) Puerto pasivo (`PASV`)

Este comando sirve para indicar al servidor que, cuando se le envíe un comando de transferencia, en lugar de establecer de manera activa la conexión de datos, debe quedar preparado para recibirla de manera pasiva. En la respuesta (código 227), el servidor devuelve la dirección en

ANOTACIONES

que esperará las peticiones de conexión, y utilizará la misma notación del argumento del comando PORT.

El comando PASV se utiliza en las transferencias entre servidores. El cliente debe establecer conexiones de control con los dos servidores, enviar un comando PASV a uno de los mismos y pasar la dirección devuelta al otro con un comando PORT. Entonces debe enviar el comando de transferencia correspondiente (leer o almacenar) al servidor pasivo, y el comando complementario al activo.

13) Reservar espacio (**ALLO**)

Algunos sistemas pueden requerir que se especifique la longitud de un fichero antes de almacenarlo. El argumento constituye el número de bytes lógicos a reservar. Si es necesario, el primer argumento puede ir seguido de la cadena R *n*, donde n indica la longitud máxima de los registros o páginas (para ficheros con tipo de estructura R o P).

Nota

Recordad que cada byte lógico tiene *n* bits, donde *n* es el argumento del comando TYPE L o, por defecto, 8.

14) Obtener (**RETR**)

Ésta es la operación de transferencia de ficheros del servidor hacia el cliente (o hacia el servidor pasivo). El argumento es el nombre del fichero que debe transferirse.

Tanto en esta operación como en las de almacenar y añadir, si el modo de transmisión es B o C, el proceso que envía los datos puede insertar un tipo especial de bloque denominado *marca de reanudación* (su contenido es un identificador de la posición actual del fichero), que deberá utilizarse en caso de error de la transferencia. Cuando encuentra la marca, el receptor asocia a la posición actual un identificador propio y se lo notifica al usuario. Si quien actúa de receptor es el servidor, activo o pasivo, la notificación se lleva a cabo por medio de una respuesta con el código 110 y el texto MARK *c* = *s*. (*c* y *s* son los identificadores de la posición del cliente y del servidor, respectivamente).

15) Almacenar (**STOR**)

Ésta es la operación de transferencia de ficheros del cliente hacia el servidor. El argumento es el nombre del fichero en que el servidor

ANOTACIONES

debe almacenar los datos. Si este fichero no existe, se crea; si ya existe, se descarta su contenido y se sustituye por los datos recibidos.

16) Almacenar con nombre único (**STOU**)

Esta operación es como la anterior, pero sin argumento: el servidor elegirá un nombre para el fichero (en el directorio actual) que no coincida con ninguno de los ya existentes. En la respuesta debe notificarse el nombre elegido.

17) Añadir (**APPE**)

Esta operación también es como la de almacenar, excepto que, si el fichero ya existe, los datos recibidos se añadirán al final de su contenido.

18) Listar (**LIST**)

El argumento de este comando es opcional. Si no hay argumento, el servidor envía por la conexión de datos una lista de los ficheros del directorio actual y sus atributos, por lo general en un formato propio del sistema. Si hay argumento, la lista se refiere al fichero o grupo de ficheros especificado, o a los ficheros contenidos en el directorio especificado.

19) Listar nombres (**NLST**)

Este comando es como el anterior, pero con el formato de la lista normalizado, lo que significa que consta sólo de los nombres de los ficheros, separados por finales de línea, que permite que la lista devuelta sea procesada, por ejemplo, para que el cliente pueda implementar una operación para obtener un grupo de ficheros.

Nota

Como precaución, el cliente se debería asegurar que el tipo de representación es A o E antes de enviar el comando LIST o NLST.

20) Abortar (**ABOR**)

Este comando hace que el servidor interrumpa la transferencia en curso (si hay) y, después, cierre la conexión de datos. Si no, simplemente cierra la conexión de datos. En el primer caso, el servidor envía un código 426 como respuesta al comando de transferencia y, a continuación, un código 226 en respuesta al comando ABOR. En el segundo caso, sólo envía la respuesta 226.

ANOTACIONES

Nota

Para indicar que el servidor debe atender un comando (ABOR, STAT, QUIT) mientras se esté llevando a cabo una transferencia, la especificación RFC 959 sugiere que el cliente envíe para la conexión de control el código de función IP del protocolo Telnet, seguido de una señal *synch* (es decir, un envío de datos urgentes TCP combinado con el código DM), y a continuación el comando FTP.

21) Retomar (**REST**)

Si una transferencia ha sido interrumpida por cualquier causa (caída del servidor, del cliente, de la red, etc.) es posible retomarla a partir de un punto indicado por una marca de reanudación. El cliente debe enviar el comando REST con un argumento igual que el identificador de posición del servidor correspondiente a la marca deseada e, inmediatamente a continuación, el comando de transferencia que desea retomar.

Nota

Recordad que sólo puede haber marcas de reanudación en los modos de transmisión B o C.

22) Borrar (**DELE**)

El efecto de este comando es borrar el fichero especificado en el argumento.

23) Nombre antiguo (**RNFR**)

Para cambiar el nombre de un fichero, en primer lugar el cliente debe enviar este comando, con el nombre actual en el argumento e, inmediatamente, el comando RNTO.

24) Nombre nuevo (**RNTO**)

El argumento de este comando es el nombre nuevo que se debe conferir al fichero. Sólo se puede utilizar inmediatamente a continuación de un comando RNFR.

25) Crear un directorio (**MKD**)

El servidor crea el directorio indicado por el argumento. Si la operación tiene éxito, el formato del código de respuesta (código 257) es idéntico al del comando PWD.

Nota

Según la especificación RFC 959, el nombre retornado en la respuesta 257 debe ser apto para poderlo utilizar como argumento del comando `CWD` (cambiar directorio). Como en algunos sistemas (obsoletos) este argumento no podía ser un nombre relativo al directorio actual, sino que debía ser un nombre absoluto (y el del comando `MKD`, en cambio, sí que podía ser relativo), por norma general los servidores responden al comando `MKD` con el nombre completo del directorio creado.

26) Borrar un directorio (**`RMD`**)

El servidor borra el directorio indicado por el argumento.

27) Estatus (**`STAT`**)

Si se envía este comando durante una transferencia, el servidor incluye en la respuesta información sobre el estado actual de la transferencia. Si no, se le puede pasar un nombre de fichero como argumento para que el servidor envíe información similar a la del comando `LIST`, pero por la conexión de control, o bien se puede utilizar sin argumento para obtener información general sobre el proceso FTP.

28) Sistema (**`SYST`**)

El servidor envía una respuesta informando sobre qué tipo de sistema es. La primera palabra del texto debe ser uno de los nombres de sistema normalizados de la lista oficial de números asignados.

Nota

La IANA (Internet Assigned Numbers Authority) es la encargada de publicar la lista de números asignados. En el momento de elaborar esta unidad, la última versión de la lista se puede encontrar en la especificación RFC 1700.
ftp://ftp.isi.edu/in-notes/iana/assignments/

Ejemplo

Un cliente Unix puede utilizar este comando para saber si el servidor también es Unix y, si lo es, invocar automáticamente el comando `TYPE I` para mejorar la eficiencia de las transmisiones.

ANOTACIONES

29) Servicios adicionales (**SITE**)

Si el servidor ofrece servicios adicionales además de las operaciones estándar del protocolo, el cliente puede invocarlos por medio de los argumentos del comando `SITE`, cuya sintaxis depende del mismo servidor.

30) Ayuda (**HELP**)

Este comando, mediante el cual el servidor envía información general sobre la implementación del protocolo (por ejemplo, qué comandos soporta y cuáles no), se puede utilizar sin argumentos. No obstante, el servidor también puede proporcionar información más específica si se le pasan argumentos (por ejemplo, un nombre de comando).

31) Operación nula (**NOOP**)

El servidor simplemente envía una respuesta 200.

32) Reinicializar (**REIN**)

El servidor reinicializa la sesión olvidando la identidad del usuario y volviendo a dar a los parámetros de transmisión los valores por defecto. La sesión queda en el mismo estado en que se encontraba justo después de establecer la conexión de control.

33) Acabar la sesión (**QUIT**)

El servidor da la sesión por finalizada y cierra la conexión de control (si hay una transferencia en curso, después de enviar la respuesta correspondiente).

La tabla siguiente resume la sintaxis de los comandos y los posibles códigos de respuesta a cada uno de los mismos según la especificación RFC 959:

Tabla 9.

Sintaxis de los comandos y códigos de respuesta	
Comando	**Respuesta**
`Establecimiento de conexión`	220, 120, 421
`USER` *`usuario`*	230, 331, 332, 530

ANOTACIONES

Sintaxis de los comandos y códigos de respuesta	
Comando	**Respuesta**
PASS *contraseña*	230, 202, 332, 530
ACCT *cuenta*	230, 202, 530
STRU F|R|P	200, 504
MODE S|B|C	200, 504
TYPE A|E|I|L [N|T|C|*n*]	200, 504
CWD *directorio*	250, 550
CDUP	250, 550
SMNT *sist-ficheros*	250, 202, 550
PWD	257, 550
PORT *h1,h2,h3,h4,p1,p2*	200
PASV	227
ALLO *long-1* [R *long-2*]	200, 202, 504
RETR *fichero*	226, 250, 110, 125, 150, 550
STOR *fichero*	226, 250, 110, 125, 150, 452, 532, 552, 553
STOU	226, 250, 110, 125, 150, 452, 532, 552, 553
APPE *fichero*	226, 250, 110, 125, 150, 452, 532, 550, 552, 553
LIST [*nombre*]	226, 250, 125, 150
NLST [*nombre*]	226, 250, 125, 150
ABOR	226
REST *marca*	350
DELE *fichero*	250, 550
RNFR *fichero*	350, 450, 550
RNTO *fichero*	250, 503, 532, 553
MKD *directorio*	257, 550
RMD *directorio*	250, 550
STAT [*nombre*]	211, 212, 213
SYST	215
SITE *cadena...*	200, 202
HELP [*cadena*]	211, 214
NOOP	200
REIN	220, 120, 421
QUIT	221

ANOTACIONES

Además de los códigos de respuesta específicos contenidos en esta tabla, el cliente también puede recibir códigos de respuesta generales a cada comando, como 500, 501, 502, 530 y 421, o bien, en el caso de los comandos de transferencia, 425, 426 y 450.

Nota

La respuesta 502 sólo es válida para los comandos no básicos; es decir, los que no pertenecen al grupo de la "implementación mínima"

18.1.4. Implementaciones del FTP

A la hora de implementar el protocolo FTP, debemos distinguir servidor de cliente:

1) Servidores FTP

Según la especificación RFC 959, la implementación mínima de un servidor FTP debe soportar los comandos siguientes:

- `USER`
- `RETR`
- `TYPE A N`
- `NOOP`
- `STOR`
- `MODE S`
- `QUIT`
- `PORT`
- `STRU F`
- `STRU R`

Nota

Muchos servidores soportan el acceso público (posiblemente restringido) a una parte de su sistema de ficheros sin necesidad de seguir el procedimiento normal de autenticación.

Este tipo de servicio se conoce como *FTP anónimo*. Por norma general, se accede al mismo utilizando el nombre de usuario `anonymous` como argumento del comando `USER`. Si el servidor pide una contraseña, suele aceptar cualquier cadena. En este caso, es costumbre proporcionar la dirección de correo electrónico del usuario como contraseña.

ANOTACIONES

2) Clientes FTP

En la actualidad, existen muchas implementaciones de clientes FTP para una gran variedad de sistemas diferentes (incluyendo los navegadores WWW); sin embargo, el cliente más utilizado durante mucho tiempo ha sido la utilidad del sistema operativo Unix y de las distribuciones GNU/Linux denominada precisamente `ftp`.

Este cliente presenta al usuario la mayoría de las respuestas del servidor, incluyendo los códigos numéricos. Los principales comandos que ofrece son los siguientes:

Nota

Recordad que los nombres de los comandos se pueden abreviar siempre que no generen ambigüedades.

- `open`: permite especificar el nombre del servidor al que es preciso conectarse, si no se ha pasado como argumento del programa, y entonces pide automáticamente el nombre de usuario y, si procede, la contraseña.
- `cd`, `pwd`, `dir`: envían los comandos `CWD`, `PWD` y `LIST` al servidor.
- `ascii`, `binary`: envían los comandos `TYPE A` y `TYPE I` al servidor.
- `get`: efectúa la secuencia `PORT-RETR` para copiar un fichero del servidor.
- `put`: efectúa la secuencia `PORT-STOR` para copiar un fichero en el servidor.
- `^C` (o el carácter que genere la señal de interrupción): envía el comando `ABOR`.
- `delete`, `mkdir`, `rmdir`: envían los comandos `DELE`, `MKD` y `RMD`.
- `rename`: envía la secuencia `RNFR-RNTO` necesaria para cambiar el nombre de un fichero.
- `mget`: envía el comando `NLST` para saber qué ficheros concuerdan con un patrón, y después una serie de comandos `RETR` para poderlos copiar.
- `mput`: envía una serie de comandos `STOR`.
- `mdelete`: envía el comando `NLST` y, a continuación, una serie de comandos `DELE`.

- `quote`: permite enviar un comando directamente al servidor (por ejemplo, `quote syst`).
- `?`: muestra un mensaje de ayuda.
- `quit` o `bye`: envía el comando `QUIT` y cierra el programa.

18.1.5. Ejemplo de sesión FTP

A continuación, mostramos los mensajes intercambiados entre un cliente y un servidor en una hipotética sesión de transferencia por medio de la utilidad `ftp`.

Después de cada comando de usuario, se muestran los mensajes enviados, y se indica su origen y el destino (`C` para el cliente, `S` para el servidor), y el número de puerto:

```
open ftp.uoc.es
C:4695 → S:21     (establecimiento de conexión)
S:21 → C:4695     220 tibet FTP server (5.6) ready.<CRLF>
(usuari) anonymous
C:4695 → S:21     USER anonymous<CRLF>
S:21 → C:4695     331 Guest login ok, send ident
                  as password.<CRLF>
(contraseña) usuari@acme.com
C:4695 → S:21     PASS usuari@acme.com<CRLF>
S:21 → C:4695     230 Guest login ok, access restrictions
                       apply.<CRLF>
cd pub
C:4695 → S:21     CWD pub<CRLF>
S:21 → C:4695     250 CWD command successful.<CRLF>
dir
C:4695 → S:21     PORT 193,146,196,254,18,88<CRLF>
S:21 → C:4695     200 PORT command successful.<CRLF>
C:4695 → S:21     LIST<CRLF>
S:20 → C:4696     (establecimiento de conexión)
S:21 → C:4695     150 ASCII data connection for /bin/ls
                  (193.146.196.254,4696) (0 bytes).<CRLF>
S:20 → C:4696     total 20<CRLF>
                  drwxr-xr-x 7 0 other 512 Aug 3 07:10
                  .<CRLF>
                  ...
S:20 → C:4696     (cierre de conexión)
S:21 → C:4695     226 ASCII Transfer complete.<CRLF>
```

```
cd doc
C:4695 → S:21     CWD doc<CRLF>
S:21 → C:4695     250 CWD command successful.<CRLF>

dir
C:4695 → S:21     PORT 193,146,196,254,18,89<CRLF>
S:21 → C:4695     200 PORT command successful.<CRLF>
C:4695 → S:21     LIST<CRLF>
S:20 → C:4697     (establecimiento de conexión)

S:21 → C:4695     150 ASCII data connection for /bin/ls
                  (193.146.196.254,4697) (0 bytes).<CRLF>
S:20 → C:4697     total 886<CRLF>
                  drwxr-xr-x 2 0 other 512 Oct 24 1996 .<CRLF>

                  ...
S:20 → C:4697     (cierre de conexión)
S:21 → C:4695     226 ASCII Transfer complete.<CRLF>
get README
C:4695 → S:21     PORT 193,146,196,254,18,91<CRLF>
S:21 → C:4695     200 PORT command successful.<CRLF>
C:4695 → S:21     RETR README<CRLF>
S:20 → C:4699     (establecimiento de conexión)
S:21 → C:4695     150 ASCII data connection for README
                  (193.146.196.254,4699) (95 bytes).<CRLF>
S:20 → C:4699     (contenido del fichero)

S:20 → C:4699     (cierre de conexión)
S:21 → C:4695     226 ASCII Transfer complete.<CRLF>

bye
C:4695 → S:21     QUIT<CRLF>
S:21 → C:4695     221 Goodbye.<CRLF>
S:21 → C:4695     (cierre de conexión)
```

18.2. El TFTP

Hay situaciones en las que se necesita transferir ficheros de un ordenador a otro y el FTP no es apropiado, principalmente a causa de su complejidad.

Nota

Un ejemplo típico en el que se observa la necesidad de transferir ficheros de un ordenador a otro es el de las estaciones de trabajo sin disco, que cargan el sistema operativo por medio de la red. En este caso, debe haber un ordenador que funcione como “servidor de arranque” de la estación, proporcionándole el fichero o ficheros en que se encuentra el código ejecutable del sistema operativo.

ANOTACIONES

Un pequeño programa residente en la memoria ROM de la estación debe controlar la transferencia de los ficheros.

Para esta operación el FTP no es adecuado, puesto que requiere una implementación del protocolo de transporte TCP, establecer en el mismo diferentes conexiones simultáneas, etc.



Para satisfacer las necesidades de transmisiones simplificadas, se ha definido el TFTP, cuya última versión está especificada en el estándar RFC 1350.

Este protocolo está basado en datagramas, sólo proporciona dos operaciones (leer y escribir ficheros) y no hay ningún tipo de identificación ni autenticación de usuario.

18.2.1. Conceptos básicos del TFTP

Como el TFTP se basa en datagramas, generalmente se utiliza con el protocolo de transporte UDP. El número de puerto al que el cliente debe enviar las peticiones de servicio es el 69.

Una **transferencia TFTP** se inicia con un datagrama enviado por el cliente en el que se solicita la operación deseada: leer o escribir un fichero. A partir de entonces, se establece un diálogo en el que cada parte envía un datagrama de manera alternada. Cada uno de estos datagramas sirve de confirmación de recepción del anterior. Ello significa que en cada momento sólo puede haber un datagrama pendiente de ser confirmado.

De este modo, la recuperación de los errores de transmisión es muy simple: si pasa un tiempo sin que llegue la respuesta a un datagrama, se reenvía y si se recibe un datagrama que ya se había recibido con anterioridad, se ignora. Por tanto, basta con guardar el último datagrama enviado por si debe retransmitirse.

Lectura complementaria

Si queréis más información sobre el TFTP, consultad la obra siguiente:

K. Sollins (1992, julio). *RFC 1350 - The TFTP protocol.*

ANOTACIONES

El cliente debe enviar su primer datagrama desde un puerto `C` al puerto 69 del servidor. Cuando lo recibe, el servidor elige un número de puerto `S` que debería ir cambiando en cada transferencia. Todos los datagramas que envíe el servidor tendrán como puerto de origen el número `S` y como puerto de destino el número `C`; todos los datagramas que envíe el cliente, excepto el primero, tendrán como puerto de origen el número `C` y como puerto de destino el número `S`. Ello permite detectar una posible duplicación del primer datagrama: si el servidor lo recibe dos veces o más, debe enviar ambas respuestas desde puertos diferentes; el cliente aceptará la primera y enviará mensajes de error a los puertos de los que provengan las otras.

En el transcurso de la transferencia, una de las partes envía los datos del fichero y la otra sólo envía confirmaciones. Los datos se envían en bloques de longitud fija, 512 bytes, excepto el último bloque, que tendrá entre 0 y 511 bytes. Cada bloque se envía en un datagrama.

Nota

Habrá un último bloque de cero bytes sólo cuando la longitud del fichero que deba transmitirse sea múltiplo de 512.

La transferencia se acaba cuando el receptor recibe el último bloque de datos y envía la confirmación correspondiente. En este momento, puede dar por finalizada la comunicación. Opcionalmente, puede esperar por si vuelve a recibir el último bloque, lo que significaría que la última confirmación no ha llegado al emisor. Si sucede esto, sólo es preciso reenviar esta confirmación.

Por su parte, el emisor dará por acabada la transferencia cuando reciba la última confirmación o cuando haya transcurrido cierto tiempo retransmitiendo el último bloque de datos y no reciba respuesta. En este último caso, podría ser que la transferencia se hubiera efectuado correctamente y que el único problema estuviera en la transmisión de la última confirmación.

18.2.2. Funcionalidad del TFTP

El TFTP ofrece dos operaciones básicas: leer ficheros del servidor y escribir ficheros en el servidor. El datagrama inicial indica la operación que el cliente quiere llevar a cabo y tiene el formato siguiente:

Figura 81.



ANOTACIONES

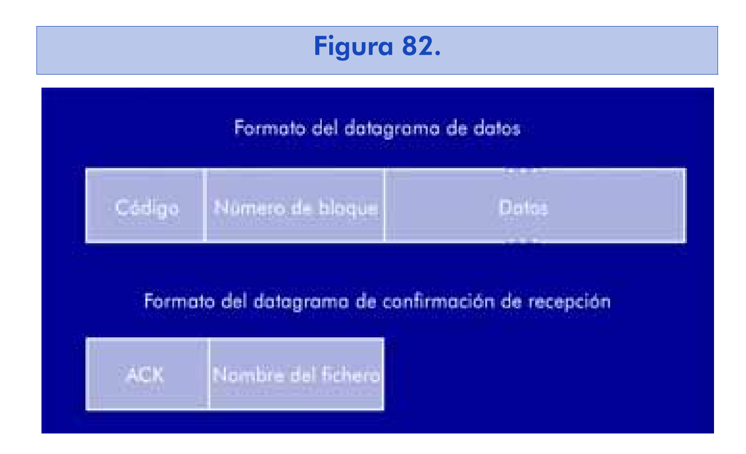
El código de operación es un número de dos bytes, que puede ser RRQ o WRQ si el cliente solicita leer o escribir un fichero, respectivamente. A continuación, existen dos cadenas de caracteres, acabadas con un byte igual a cero: la primera es el nombre del fichero y la segunda es el modo de transferencia (el equivalente del tipo de representación en FTP). Esta segunda cadena puede ser netascii u octet (en caracteres ASCII, y en cualquier combinación de mayúsculas y minúsculas). El primer valor indica que los datos son caracteres ASCII tal como se usan en el protocolo Telnet, y el segundo indica que los datos son bytes arbitrarios de 8 bits.

Nota

En versiones anteriores del protocolo, había un tercer modo de transferencia llamado mail, sólo aplicable a las operaciones de escritura, en el que el nombre del fichero era sustituido por el nombre de un usuario que debía recibir los datos por correo.

Los datagramas que contienen los datos y las confirmaciones de recepción tienen los formatos siguientes:

Figura 82.



El primer campo es el código de operación y el segundo es el número de bloque que se envía o se confirma (ambos campos son de dos bytes). Cada bloque del fichero tiene un número correlativo, empezando por 1, que sirve para distinguir las confirmaciones duplicadas.

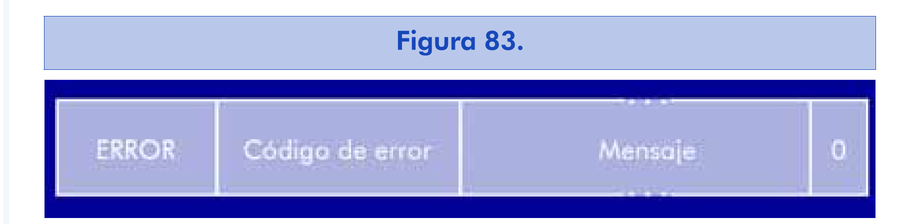
Si el cliente envía un datagrama inicial RRQ, el servidor contesta con un datagrama DATA con número de bloque igual a 1 y, si el cliente envía un datagrama inicial WRQ, el servidor contesta con un datagrama ACK con número de bloque igual a 0 y, a continuación, el cliente

ANOTACIONES

envía el primer datagrama de datos. Entonces, cliente y servidor se intercambian datagramas `ACK` y `DATA` de manera alternada, con las retransmisiones necesarias si no llega el datagrama que corresponde en cada momento.

El TFTP también prevé la terminación de la transferencia si se produce algún error. Cuando se detecta el error, se envía un datagrama con el formato siguiente:

Figura 83.



Los dos primeros campos son el código de operación y el código de error (cada uno de dos bytes). A continuación, hay una cadena de caracteres, acabada en 0, que puede servir para describir a un usuario humano la causa del error.

Nota

Si llega un datagrama del servidor con un número de puerto de origen incorrecto (probablemente a causa de un datagrama inicial duplicado), la transferencia con este puerto queda interrumpida, pero la que utiliza el puerto correcto debe continuar con normalidad.

Un datagrama de error indica que se da por acabada la transferencia y no debe confirmarse ni, por tanto, retransmitirse. Ahora bien, si por alguna razón se pierde este datagrama, la otra parte interpretará que la transferencia ha acabado prematuramente cuando haya transcurrido cierto tiempo retransmitiendo sin recibir nada.

En las tablas siguientes se presenta una relación de los códigos numéricos asignados a cada operación y a cada tipo de error TFTP:

Tablas 10 y 11.

Código	Operación
1	RRQ
2	WRQ
3	DATA
4	ACK
5	ERROR

Código	Error
0	Error indefinido (veáse el mensaje)
1	No se ha encontrado el fichero
2	Acceso denegado
3	Disco lleno
4	Operación no válida
5	Número de puerto incorrecto
6	Ya existe el fichero
7	Usuario incorrecto (en modo `mail`)

ANOTACIONES

18.2.3. Implementaciones del TFTP

En muchos sistemas Unix y distribuciones GNU/Linux hay un servidor del TFTP llamado `tftpd`. Como no existe ningún tipo de identificación de usuario, los clientes pueden acceder en principio a cualquier fichero con permisos de acceso público. En algunos sistemas, sin embargo, puede restringirse el acceso a un directorio o unos directorios determinados. No obstante, el servicio TFTP suele estar inhabilitado, y sólo lo ofrecen los sistemas en que se necesita por algún motivo concreto (por ejemplo, los servidores de arranque de las estaciones sin disco).

Asimismo, hay un programa cliente llamado `tftp` que funciona de manera similar a la utilidad `ftp`: admite comandos como `get`, `put`, `ascii`, `binary` o `quit` (pero no `cd`, `dir`, `delete`, `rename`, etc., porque el protocolo no los soporta).

Nota

Por norma general hay un directorio llamado `/tftpboot` en el que se guardan las imágenes de los sistemas operativos de los clientes sin disco, y sólo se permite a este directorio el acceso por medio del TFTP.

ANOTACIONES

19. Correo electrónico Internet



El correo electrónico es la aplicación distribuida que permite enviar mensajes electrónicos por medio de sistemas informáticos.

Al especificar esta aplicación, se consideró adecuado que todas sus características siguieran las ideas básicas del correo postal:

- Las **operaciones** permiten básicamente enviar mensajes y recibirlos.

- Los **elementos** son equivalentes a los mensajes, los usuarios, las oficinas de correo y los buzones.

- La **funcionalidad** está basada en la filosofía del almacenamiento y el reenvío: cuando un mensaje llega a una oficina de correos, esta última lo almacena y no lo reenvía hasta el momento en que lo considera oportuno. De este modo, los mensajes van de oficina de correos en oficina de correos hasta que llegan al destinatario.

Nota

La filosofía del almacenamiento y el reenvío permite superar problemas como las caídas de la red; en este caso, los mensajes no se pierden, puesto que en cada momento hay una oficina responsable del mensaje.

Para llevar a cabo esta funcionalidad, se definieron los protocolos siguientes:

- **SMTP** (*simple mail transfer protocol*), para la **transferencia de mensajes**.

- **POP3** (*post office protocol*), para el **acceso simple a buzones de correo**.

- **IMAP4rev1** (*Internet message access protocol*), para el **acceso complejo a buzones de correo**.

También fue necesario definir un **formato de mensaje**, el **RFC 822**, que con posterioridad se amplió y dio lugar al formato **MIME**.

ANOTACIONES

19.1. Formato de los mensajes: el RFC 822

Lectura complementaria

Si deseáis más información sobre el formato de los mensajes de correo Internet, el RFC 822, consultad la obra siguiente:

D. Crocker (1982, 13 de agosto). *RFC 822 - Standard for the format of ARPA Internet text messages*.

Nota

Consultad en el anexo 3 la notación que se utiliza para describir los campos de los mensajes.

Antes de describir los diferentes protocolos relacionados con el correo electrónico, es preciso ver cuál es el formato o la norma de los mensajes que se utiliza en el mismo. Este formato recibe el nombre del estándar en que se especifica, **RFC 822**, y se basa en los elementos típicos de las cartas postales; es decir, el sobre, que contiene la información del destinatario y del remitente de la carta, el contenido, que es el mensaje en sí.

El estándar especifica que los mensajes de correo electrónico están formados por las dos partes siguientes:

- La **cabecera**, que recoge la información general del mensaje. Equivale al sobre de la carta postal y está formada por una serie de **campos de cabecera**, cada uno de los cuales incluye un tipo concreto de información.

- El **cuerpo del mensaje**, que contiene el mensaje en sí. Corresponde al contenido de la carta postal. Esta parte es opcional.

Cada campo de cabecera consta de un **nombre del campo** (que identifica el tipo de información) seguido por el carácter ":", opcionalmente acompañado por un **cuerpo del campo** (que incluye la información del campo), y acaba con un carácter `<CRLF>`. El formato de los campos es el siguiente:

```
NombredelCampo:[CuerpodelCampo]<CRLF>
```

Por norma general, cada campo de cabecera se representa en una sola línea, empezando desde el primer carácter de la misma. En el caso de que se necesite más de una, es preciso codificar estas líneas adicionales empezando por un carácter, o más, de espacio o tabulador.

El cuerpo del mensaje es simplemente una secuencia de líneas con caracteres ASCII. El cuerpo está separado de la cabecera por una línea nula (es decir, por la secuencia `<CRLF><CRLF>`).

ANOTACIONES

Algunos sistemas de correo electrónico permiten reenviar mensajes a los usuarios. Por este motivo, se incluyen en el mismo unos campos de cabecera con información sobre el reenvío. Estos campos son los que llevan el prefijo `Resent-` y tienen la misma semántica que los campos correspondientes que no lo llevan. No obstante, siempre convendrá tener presente que, dentro de un mensaje, los campos que llevan prefijo son los más recientes.

19.1.1. Información de la cabecera

Los campos de cabecera del mensaje deben proporcionar información general del mensaje, incluyendo la información equivalente a la de un sobre postal; de este modo, encontramos los campos siguientes:

Nota

Consultad el formato de las direcciones para identificar los buzones de usuario en el apartado 19.2.2.

1) Originador (**`From/Resent-From`**)

La identidad y la dirección del buzón de la persona o las personas que originan el mensaje se puede incluir en el campo `From` o `Resent-From`. En este campo, se puede introducir la dirección del buzón de un originador o más.

```
From: 1#dirección
Resent-From: 1#dirección
```

2) Destinatario (**`To/Resent-To`**)

El RFC 822 identifica al destinatario o destinatarios principales del mensaje por medio del campo `To` o `Resent-To`. En este campo, se puede introducir la dirección de un destinatario o más.

```
To: 1#dirección
Resent-To: 1#dirección
```

3) Destinatario de copia (**`Cc/Resent-cc`**)

Asimismo, existe la posibilidad de enviar copias de un mensaje. En este caso, la identidad del receptor o receptores secundarios del mensaje

ANOTACIONES

se especifica con el campo `Cc` o `Resent-cc`. En este último, se puede introducir la dirección de un destinatario de copia o más.

```
Cc: 1#dirección
Resent-cc: 1#dirección
```

4) Destinatario adicional (o de copia ciega) (**`Bcc/Resent-bcc`**)

Cuando se desea enviar una copia del mensaje a destinatarios adicionales, sin que ninguno de ellos (los principales, los de copia y los de copia ciega) sepan que otros destinatarios la han recibido, se utiliza el campo `Bcc` o `Resent-bcc` que no lo ve ninguno de ellos.

```
Bcc: 1#dirección
Resent-bcc: 1#dirección
```

5) Destinatario de respuesta (**`Reply-To/Resent-Reply-To`**)

La identificación del destinatario o destinatarios a los que deben enviarse las respuestas se puede llevar a cabo por medio del campo `Reply-To` o `Resent-Reply-To`; en este último se puede introducir la dirección de un destinatario de la respuesta o más.

```
Reply-To: 1#dirección
Resent-Reply-To:1# dirección
```

6) Asunto (**`Subject`**)

Otra posibilidad que ofrece el RFC 822 es enviar un texto explicativo del asunto del mensaje con el campo `Subject`.

```
Subject: texto
```

7) Data (**`Date/Resent-Date`**)

Dentro de un mensaje también puede incluirse la hora y la fecha en que se envía por medio del campo `Date` o `Resent-Date`. El primer

ANOTACIONES

sistema de correo que recibe el mensaje genera automáticamente este campo.

```
Date: fecha-hora
Resent-Date: fecha-hora
```

8) Sistema remitente (**Sender/Resent-Sender**)

Algunas veces el usuario que envía el mensaje no es el mismo que el autor. En estos casos, la identidad del agente (persona, sistema o proceso) que envía en realidad el mensaje se identifica con el campo `Sender` o `Resent-Sender`.

```
Sender: buzón
Resent-Sender: buzón
```

9) Camino de retorno hacia el originador (**Return-path**)

El mensaje puede incluir la identificación del camino de retorno hacia el originador del mensaje con el campo `Return-path`. Este campo lo añade el sistema de transporte final que entrega el mensaje al receptor.

```
Return-path: addr-ruta
```

10) Información de sistemas intermedios (**Received**)

Existe la posibilidad de que cada sistema de transporte intermedio por el cual pasa el mensaje incluya información de los sistemas emisor (`from`) y receptor (`by`), del mecanismo físico (`via`), del identificador del mensaje (`id`), de las especificaciones originales (`for`) y de la hora de recepción por medio del campo `Received`. Cada sistema intermedio añade una copia de este campo al mensaje.

```
Received:
 [from dominio]
 [by dominio]
 [via atom]
 *(with atom)
[id id-msg]
 [for addr-spec]
 ; fecha-hora
```

ANOTACIONES

11) Identificador del mensaje (**Message-ID/Resent-Message-ID**)

Cada mensaje debe incluir un identificador único del mensaje, para que pueda ser contestado o referenciado con posterioridad, en el campo `Message-ID` o `Resent-Message-ID`. El sistema que genera el identificador es el encargado de asegurar que el identificador sea único.

```
Message-ID: id-msg
Resent-Message-ID: id-msg
```

12) Identificador del mensaje contestado (**In-Reply-To**)

Cuando un mensaje constituye la respuesta de un mensaje anterior, el mensaje original se puede referenciar por medio de su identificador dentro del campo `In-Reply-To`.

```
In-Reply-To:
  *(frase | id-msg)
```

13) Identificador de mensajes referenciados (**References**)

Si se desea enviar un mensaje que se refiere a otros mensajes, el identificador del mensaje referenciado se puede incluir dentro del campo `References`.

```
References:
  *(frase | id-msg)
```

14) Palabras clave (**Keywords**)

Las palabras clave o frases referentes al mensaje se pueden incluir, separadas por comas, dentro del campo `Keywords`.

```
Keywords: #frase
```

15) Comentarios (**Comments**)

Se puede incluir un texto de comentario en el mensaje, sin interferir su contenido, por medio del campo `Comments`.

```
Comments: texto
```

ANOTACIONES

16) **Identificación del mecanismo de cifrado** (`Encrypted`)

El RFC 822 permite cifrar el cuerpo del mensaje. En este caso, es conveniente enviar una o dos palabras que identifiquen el mecanismo de cifrado que se ha aplicado utilizando el campo `Encrypted`.

```
Encrypted: 1#2palabras
```

Nota

El RFC 822 sólo define la sintaxis para especificar un mecanismo de cifrado; sin embargo, no especifica el mecanismo ni la manera de utilizarlo.

17) **Campos de uso personal.**

El estándar permite también que los usuarios definan campos para uso personal. El nombre de los campos creados por un usuario debe empezar obligatoriamente por la cadena `X-`.

```
X-campo-uso-personal
```

18) **Campos de extensión**

En el futuro se pueden estandarizar nuevos campos de cabecera. Para que no haya conflictos con los campos de uso personal, el nombre de los campos nunca empezará por la cadena `X-`.

```
campo-extensión
```



El estándar RFC 822 establece que los únicos campos de cabecera que son obligatorios en un mensaje son los siguientes: fecha (`Date`), originadores (`From`), y destinatario (`To`) o destinatario de copia ciega (`Bcc`).

ANOTACIONES

19.1.2. Ejemplo

En este apartado se presenta un ejemplo de mensaje RFC 822 en el que se pueden apreciar los campos de cabecera más típicos, así como un cuerpo breve del mensaje. Conviene recordar que es un mensaje en ASCII de 7 bits.

```
Date: 25 Jun 2003 0932 PDT
From: Jordi Inyigo <jinyigo@uoc.edu>
Subject: Ejemplo mensaje RFC 822
Sender: jmmarques@uoc.edu
Reply-To: jinyigo@uoc.edu
To: Ramon Marti <rmarti@uoc.edu>,
    xperramon@uoc.edu
Cc: Llorenc Cerda <lcerda@uoc.edu>,
    Jose Barcelo <jbarcelo@uoc.edu>,
    epeig@uoc.edu
Comment: os envío esta información que os puede interesar
In-Reply-To: <1234321.567898765@uoc.edu>
Received: from uoc.edu by peru.uoc.es
      (8.8.5/8.8.5) with ESMTP id SAA14826
      for <rmarti@uoc.edu >; Fri, 20 Jun 2003
      18:35:52 +0200 (MET DST)
Received: from rectorat.uoc.edu(147.83.35.35)
      by uoc.es via smap (V2.0) id xma020193;
      Mon, 20 Jun 2003 18:38:50 +0200 for
      rmarti@uoc.edu
Message-Id: <199809211639.SAA20364@uoc.edu>

Este mensaje tiene formato RFC 822 e incluye
algunos de los campos de cabecera mas utilizados.
```

Actividad

Coged un mensaje de correo electrónico que hayáis recibido y analizad toda la cabecera. Identificad todos los campos y tratad de averiguar quién es el responsable de crear cada uno de los campos.

19.2. El SMTP

Lectura complementaria

Si queréis más información sobre el SMTP, consultad la obra siguiente:

J. Postel (1982, 1 de agosto). *RFC 821 - Simple Mail Transfer Protocol*.

El SMTP es el protocolo más utilizado en Internet para transferir mensajes de correo electrónico. Proporciona la funcionalidad necesaria para conseguir una transferencia fiable y eficiente de mensajes de correo entre ordenadores que actúan como oficina de correos. Siguiendo las ideas del correo postal, el SMTP se basa en el almacenamiento y el reenvío. Es decir, cuando un mensaje llega a una oficina, queda almacenado en la misma cierto tiempo antes de ser entregado a otra oficina o al destinatario final. Conviene señalar, asimismo, que cada usuario debe disponer de un buzón para recibir mensajes, el cual siempre debe estar asociado a una oficina determinada.

ANOTACIONES

19.2.1. Modelo del SMTP

Desde el punto de vista del modelo, el SMTP debe proporcionar los elementos necesarios para la transferencia de mensajes. Por ello, se definen los elementos siguientes:

- **Agente de usuario**: se encarga de introducir los mensajes en el sistema de correo SMTP.
- **Emisor SMTP**: se ocupa de realizar las conexiones y de enviar mensajes a receptores SMTP a partir de peticiones de los usuarios. Generalmente, cada emisor SMTP tiene asociada una **cola de mensajes**, en la que se almacenan los mensajes antes de ser reenviados (siguiendo la filosofía del almacenamiento y el reenvío).
- **Receptor SMTP**: se encarga de recibir los mensajes. Si el mensaje va destinado a un usuario asociado al mismo sistema en que se encuentra el receptor SMTP, éste deposita el mensaje en el buzón del destinatario. En caso contrario, el receptor SMTP deposita el mensaje en la cola de mensajes del emisor SMTP asociado, que lo recupera y lo reenvía hacia el receptor SMTP de otra oficina más próxima al destinatario.

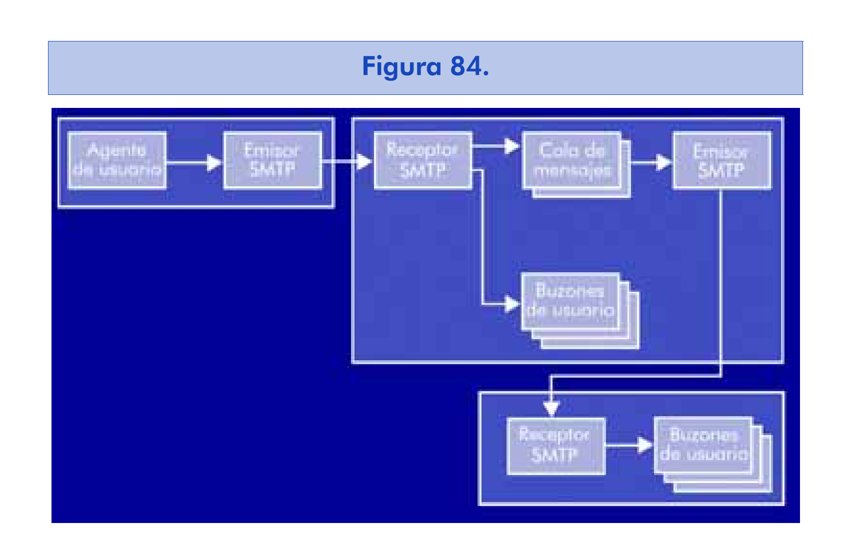
Nota

El agente de usuario, definido en la mayoría de los estándares, es equivalente al elemento usuario especificado en el modelo cliente/servidor. Puede ser tanto otra aplicación, como una persona por medio de una interfaz de usuario.

Conviene destacar que los sistemas que actúan como oficina deben disponer al mismo tiempo de un receptor SMTP (para recibir los mensajes), de buzones de los usuarios y de un emisor SMTP (para poder reenviar los mensajes) con una cola de mensajes.

La figura siguiente nos muestra el modelo de un sistema SMTP:

Figura 84.



ANOTACIONES

Nota

Notad que este modelo, así como los otros que se estudiarán en este módulo, sigue el modelo cliente/servidor definido con anterioridad. Asimismo, conviene remarcar que cada estándar proporciona un nombre diferente a los elementos del modelo cliente/servidor. En el estándard que nos ocupa, el usuario equivale al agente de usuario, el cliente equivale al emisor SMTP y el servidor equivale al receptor SMTP.

19.2.2. Direcciones de correo

Para que el sistema de correo sea capaz de entregar un mensaje, se precisa algún mecanismo que permita definir direcciones para los buzones de los usuarios. En los protocolos Internet, la **dirección de buzón** está formada por una cadena que identifica a un usuario (persona, sistema o proceso) y una cadena que identifica el sistema (dominio) en que se encuentra el buzón.

Nota

El nombre de dominio suele estar formado por la secuencia de nombres de los subdominios de los que depende jerárquicamente separados por el carácter ".".

```
dirección = usuario@dominio
dominio = subdominio*(.subdominio)
```

Con este tipo de direccionamiento electrónico, se tiene la funcionalidad siguiente:

- El mensaje se envía al sistema identificado por el nombre de dominio que se encuentra en la dirección a la derecha del signo `@` (es decir, `dominio`).

- Una vez en el sistema, el mensaje se entrega al buzón del usuario identificado en la dirección a la izquierda del signo `@` (es decir, `usuario`).

- El SMTP proporciona también la posibilidad de definir **listas de correo**, que constituyen listas de destinatarios identificadas por una única dirección. Esta última es la que se utiliza para enviar mensajes a la lista, y el sistema SMTP se encarga de expandirla y enviar el mensaje a todos los usuarios que sean miembros de la

Nota

Cada ordenador que actúa como oficina de correos suele definir un dominio de correo, que se identifica con un nombre de dominio. Este nombre es el que se encuentra en la parte *dominio* de la dirección electrónica de los usuarios que tienen los buzones en estas máquinas.

misma en aquel momento. Este método permite la modificación de la lista sin necesidad de cambiar la dirección.

19.2.3. Envío de correo y mensajes a terminales

Además del envío de mensajes de correo a buzones (en el estándar, `mail`), que es su propósito principal, el SMTP también soporta la entrega de mensajes al terminal del destinatario (en el estándar, `send`). Como la implementación de estos dos métodos es muy similar, el SMTP define comandos en que combina.

19.2.4. Conceptos básicos del SMTP

El SMTP está basado en conexiones TCP y el puerto que tiene asignado es el 25.

Como hemos comentado al describir el modelo, el emisor SMTP hace llegar mensajes de correo al receptor. Para conseguirlo, se establece un diálogo entre los dos con **comandos** que envía al emisor y **respuestas** con las que contesta el receptor.

Tanto los comandos como las respuestas SMTP siguen las reglas específicas del protocolo Telnet. Es decir, constituyen cadenas de caracteres codificados con bytes según el código ASCII y se utiliza la secuencia <CRLF> para representar el final de línea.

- Los comandos SMTP están formados por un código constituido por cuatro caracteres alfanuméricos y, según los comandos, un espacio y una serie de parámetros.
- Las respuestas SMTP están formadas por un código numérico de tres dígitos que, habitualmente, va seguido de un texto descriptivo.

En los apartados siguientes se describirán los comandos definidos y las posibles respuestas.

19.2.5. Funcionalidad del SMTP

Es preciso diferenciar entre funcionalidad básica y funcionalidad adicional.

ANOTACIONES

ANOTACIONES

Funcionalidad básica

Una vez conectado, el emisor SMTP se identifica ante el receptor SMTP con el comando HELO.

```
HELO dominio
```

Cuando se quiere iniciar el envío de un mensaje de correo, se utiliza el comando MAIL, que incluye la identificación del sistema desde el que se envía el mensaje. Este comando da lugar al campo FROM: del mensaje.

```
MAIL FROM: originador
```

Con el comando RCPT se identifican los receptores del mensaje. Se debe utilizar uno para cada receptor, y cada llamada da lugar a un campo de cabecera TO: en el mensaje.

```
RCPT TO: receptor
```

El comando DATA indica el inicio del envío del cuerpo del mensaje. Las líneas siguientes a este comando se tratan como contenido del mensaje. Este último se acaba con una línea que sólo incluye un punto; es decir, con la secuencia <CRLF>.<CRLF>.

```
DATA
```

Los datos que se envían dentro de este campo son mensajes RFC 822, por lo que pueden incluir campos de cabecera en el inicio. Cuando ello sucede, entre los campos de cabecera y el cuerpo del mensaje debe haber una línea en blanco, es decir, la secuencia <CRLF><CRLF>.

Una vez iniciada la transacción de envío de mensaje, y antes de acabar, el emisor SMTP siempre puede interrumpirla por medio del comando RSET.

```
RSET
```

El comando NOOP sirve para que el receptor SMTP envíe una respuesta afirmativa para informar de que la conexión todavía está abierta.

```
NOOP
```

Para cerrar el canal de transmisión, el SMTP proporciona el comando QUIT. Cuando este último llega al receptor SMTP, éste envía una respuesta afirmativa y cierra el canal de transmisión.

```
QUIT
```

Funcionalidad adicional

El protocolo STMP posibilita también una funcionalidad adicional con el objetivo de enviar mensajes a terminales.

Para iniciar la entrega de un mensaje, o varios, a terminales, el emisor SMTP dispone del comando SEND.

```
SEND FROM: originador
```

El comando SOML permite iniciar la entrega de un mensaje, o varios, a terminales si el usuario se encuentra en el terminal, o, en caso contrario, permite entregar el correo al buzón o a los buzones.

```
SOML FROM: originador
```

El comando SAML permite iniciar la entrega de un mensaje, o varios, a terminales y, al mismo tiempo, al buzón o a los buzones.

```
SAML FROM: originador
```

El emisor SMTP también puede pedir la confirmación de que una cadena identifica a un usuario por medio del comando VRFY.

```
VRFY cadena
```

ANOTACIONES

El comando `EXPN` permite solicitar confirmación para una cadena que identifica una lista de correo y, en caso de respuesta positiva, requerir la relación de los miembros de la lista.

```
EXPN cadena
```

El comando `HELP` permite pedir ayuda al receptor SMTP sobre los comandos que soporta. En caso de incluir una cadena como argumento, esta última puede identificar un comando, del que se retorna información específica.

```
HELP [cadena]
```

El comando `TURN` permite cambiar el papel de emisor SMTP y receptor SMTP. El resultado de la petición puede ser que el receptor SMTP envíe una respuesta afirmativa y haga el papel de emisor SMTP, o que envíe una respuesta negativa y mantenga su papel.

```
TURN
```

19.2.6. Códigos de respuesta

El SMTP define una serie de códigos de respuesta para los diferentes comandos, tanto en caso de éxito (`E`), como en caso de resultado intermedio (`I`), de fallo (`F`) y de error (`X`). La tabla siguiente recoge estos códigos:

Tabla 12.

Codigos de respuesta

	Establecimiento de connexión	HELO	MAIL	RCPT	DATA	RSET	SEND	SOML	SAML	VRFY	EXPN	HELP	NOOP	QUIT	TURN
211 Estatus del sistema o respuesta a petición de ayuda												E			
214 Mensaje de ayuda												E			
220 El servidor está preparado	E														

Codigos de respuesta															
	Establecimiento de connexión	HELO	MAIL	RCPT	DATA	RSET	SEND	SOML	SAML	VRFY	EXPN	HELP	NOOP	QUIT	TURN
221 El servidor está cerrando el canal de transmisión														E	
250 Acción completada correctamente		E	E	E	E	E	E	E	E	E	E	E	E		E
251 Usuario no local, el mensaje se reenviará				E						E					
354 Principio de entrada de mensaje; es preciso acabar con <CRLF>.<CRLF>					I										
421 Servicio no disponible; cierre del canal de transmisión	F	X	X	X	X	X	X	X	X	X	X		X		
450 Acción no ejecutada: buzón no accesible				F											
451 Acción abortada por error local			F	F	F		F	F	F						
452 Acción abortada por capacidad de almacenamiento insuficiente			F	F	F		F	F	F						
500 Error de sintaxis; comando no reconocido		X	X	X	X	X	X	X	X	X	X	X	X	X	X
501 Error de sintaxis en los parámetros o argumentos		X	X	X	X	X	X	X	X	X	X	X			
502 Comando no implementado							X	X	X	X	X	X			F
503 Secuencia de comandos errónea				X	X										X
504 Parámetro de comando no implementado		X				X				X	X	X			
550 Acción no ejecutada: buzón no encontrado				F						F	F				
551 Usuario no local; intentar otra dirección				F						F					
552 Acción abortada por exceso de capacidad almacenada			F	F	F		F	F	F						
553 Acción no ejecutada; nombre de buzón no permitido				F						F					
554 Fallo en la transacción					F										



El estándar establece el conjunto de comandos mínimo que deben soportar todos los receptores SMTP:

- `HELO` *dominio*
- `MAIL FROM:` *originador*
- `RCPT TO:` *receptor*
- `DATA`
- `RSET`
- `NOOP`
- `QUIT`

19.2.7. Extensiones SMTP para mensajes de 8 bits

Se consideró oportuno añadir un mecanismo para extender el SMTP. Cuando un cliente SMTP desea utilizar las extensiones SMTP, así como las extensiones para mensajes de 8 bits, tiene que solicitarlo al receptor SMTP con el comando `EHLO` en lugar del comando `HELO`:

```
EHLO dominio
```

Si el receptor SMTP soporta las extensiones, retorna una respuesta o más de éxito (250) e indica las extensiones de fallo (550) o de error (501) que soporta. Si el receptor no las soporta, retorna una respuesta de error (500, 502, 504 ó 421).

Lectura complementaria

Para saber más sobre las extensiones SMTP para mensajes de 8 bits, consultad las obras siguientes:

J. Klensin; N. Freed; M. Rose; E. Stefferud; D. Crocker (1995, noviembre). *RFC 1869 - SMTP Service Extensions*.

J. Klensin; N. Freed; M. Rose; E. Stefferud; D. Crocker (1994, julio). *RFC 1652 - SMTP Service Extension for 8bit-MIME transport*.

Una de las funcionalidades adicionales es la posibilidad de enviar mensajes de 8 bits. Cuando el servidor acepta mensajes de 8 bits, la respuesta de éxito (250) incluye la cadena `8BITMIME`. Cuando se quiera enviar el mensaje, debe indicarse que es de 8 bits. Ello se aplica a los comandos `MAIL`, `SEND`, `SAML` o `SOML` de la manera siguiente:

```
MAIL FROM: originador BODY = valor-cuerpo
SEND FROM: originador BODY = valor-cuerpo
SAML FROM: originador BODY = valor-cuerpo
SOML FROM: originador BODY = valor-cuerpo
valor-cuerpo = 7BIT | 8BITMIME
```

19.2.8. Ejemplo

En este apartado se presenta un ejemplo en el que se puede observar la secuencia de comandos que envía un emisor SMTP (en negrita) y las respuestas del receptor SMTP en una transacción de envío de correo:

```
220 peru.uoc.es SMTP/smap Ready.
HELO campus.uoc.es
250 (campus.uoc.es) pleased to meet you.
HELP
214-Commands
214-HELO    MAIL    RCPT    DATA    RSET
214 NOOP    QUIT    HELP    VRFY    EXPN
NOOP
220 OK
EXPN xc@campus.uoc.es
250-Jordi Inyigo <jinyigo@uoc.edu>
250-Jose Barcelo<jbarcelo@uoc.edu>
250-Llorenc Cerda <lcerda@uoc.edu>
250-Ramon Marti <rmarti@uoc.edu>
250-Enric Peig <epeig@uoc.edu>
250 Xavier Perramon <xperramon@uoc.edu>
MAIL FROM: jinyigo@uoc.edu
250 jinyigo@uoc.edu... Sender OK
RCPT TO: rmarti@uoc.edu
250 rmarti@uoc.edu OK
RCPT TO: rmarti@uoc.edu
501 Syntax error
RCPT TO: xperramon@uoc.edu
250 xperramon@uoc.edu OK

DATA
354 Enter mail, end with "." on a line by itself
Subject: Master de software libre
Date: 20 Jun 2003
Esto es un mensaje de correo de ejemplo.
.
250 Mail accepted
QUIT
250 Closing connection
```

Nota

Telnet como cliente genérico de los protocolos

El programa `telnet` está pensado para hacer de cliente de servidores del protocolo Telnet. Sin embargo, si se considera que lo que hace es enviar al servidor cadenas de caracteres tal como se introducen por el teclado y devolver por pantalla lo que recibe del servidor, también se puede utilizar como cliente SMTP puesto que, como hemos comentado al describirlo, los mensajes que se intercambian en este protocolo son cadenas de caracteres ASCII. Este mismo razonamiento se puede aplicar al resto de protocolos que veremos: POP3, IMAP, NNTP y HTTP.

De este modo, si se utiliza el programa telnet como cliente SMTP, se pueden enviar comandos SMTP al servidor escribiéndolos por el teclado y viendo sus respuestas por pantalla. Para ello, el programa telnet admite como segundo parámetro el número de puerto donde debe conectarse.

Por ejemplo, el diálogo anterior se podría haber realizado haciendo:

```
telnet peru.uoc.es 25
```

19.3. Acceso simple a los buzones de correo: el POP3

En sistemas pequeños no es práctico, ni usual, soportar el SMTP, puesto que implica tener el sistema conectado y dispuesto a recibir mensajes en cualquier momento. Por este motivo, se vio la necesidad de definir un protocolo que permitiera la recuperación de mensajes de buzones de correo remotos y se definió el **POP3**.

Nota

El POP y el POP2, dos protocolos definidos en las RFC 918 y RFC 937 respectivamente, parten de la misma filosofía que el POP3; sin embargo, disponían de un conjunto de comandos más reducido.

ANOTACIONES

No obstante, en este protocolo es preciso disponer de sistemas en los que se encuentren los buzones –un servidor POP3–, y deben estar conectados en todo momento, tanto para recibir los mensajes, como para recibir las peticiones de acceso a los buzones. Por lo que respecta a los clientes POP3, sólo es necesario que se conecten cuando quieran acceder a su correo.

El POP3 no especifica ningún método para el envío de correo; otros protocolos de transferencia de correo, como el SMTP, proporcionan esta funcionalidad.

Lectura complementaria

Si queréis más información sobre el SMTP, consultad la obra siguiente:

J. Myers; M. Rose (1996, mayo). *RFC 1939 - Post Office Protocol - Version 3*

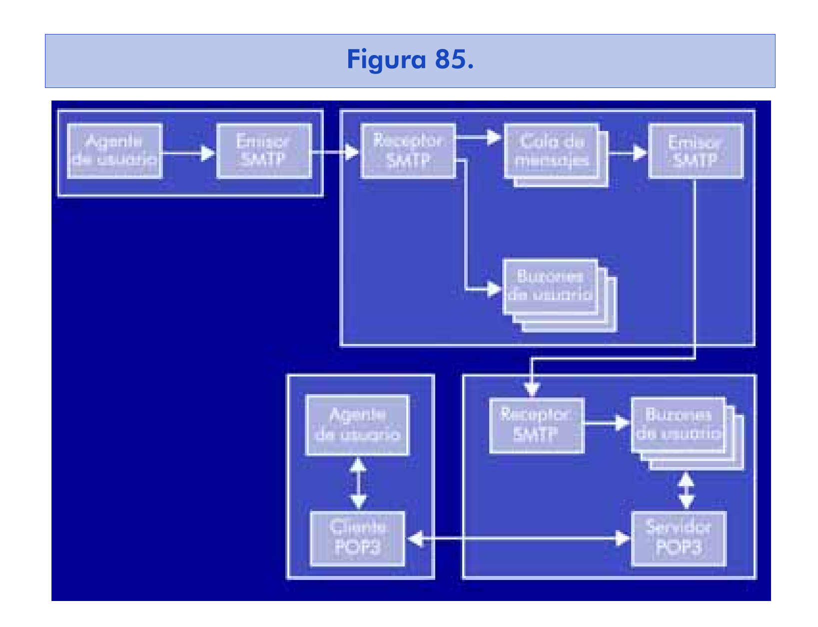
19.3.1. Modelo del POP3

El modelo funcional del POP3 se basa en los elementos siguientes:

- **Agente de usuario**: utiliza el cliente POP3 para acceder a su correo.
- **Cliente POP3**: se comunica con el servidor POP3 por medio del protocolo POP3 para acceder a su buzón de correo.
- **Servidor POP3**: recibe peticiones de los clientes POP3 y se las sirve accediendo a los buzones correspondientes.

En la figura siguiente se presentan los elementos del modelo funcional del POP3 integrados en un sistema en el que se utiliza el SMTP para enviar el correo, y el POP3 para acceder a los buzones:

Figura 85.



19.3.2. Conceptos básicos del POP3

El POP3 se basa en comunicaciones TCP sobre el puerto 110.

El mecanismo normal de utilización de dicho protocolo es el siguiente: cuando el cliente POP3 necesita acceder al buzón, se conecta con el servidor POP3, recupera la información que le interesa y cierra la conexión.

Cada vez que sea necesario volver a acceder al buzón, se establece una nueva conexión.

- Los **comandos POP3** constituyen cadenas de caracteres ASCII imprimibles acabados con `<CRLF>`. Todos los comandos incluyen un código alfanumérico de cuatro caracteres que identifica el comando, seguido de cero o más parámetros.
- Las **respuestas POP3** también son cadenas de caracteres ASCII, y se representan con un indicador de estado positivo (`+OK`) o negativo (`-ERR`) y, posiblemente, información adicional, de la manera siguiente:

```
+OK el comando se ha ejecutado con éxito
-ERR el comando no se ha ejecutado con éxito
```

Estados

La norma define tres estados por los que debe pasar toda sesión POP3:

- Una vez se ha abierto la conexión, la sesión entra en el **estado de autorización**, en que el cliente debe identificarse ante el servidor POP3.
- Una vez autorizado, la sesión pasa al **estado de transacción**. En este último, el cliente pide acciones al servidor POP3 con los comandos necesarios.
- Cuando el cliente llama el comando `QUIT`, la sesión entra en el **estado de actualización**. El servidor libera los recursos, se despide y cierra la conexión TCP.

ANOTACIONES

19.3.3. Funcionalidad del POP3

Desde el punto de vista de funcionalidad, y retomando los planteamientos del correo postal, el POP3 proporciona los comandos necesarios para el acceso a buzones de correo. A continuación, describiremos los comandos correspondientes a cada uno de los estados por los que debe pasar una sesión POP3:

1) **Estado de autorización**. En este estado el cliente se identifica ante el servidor POP3. Para realizar el proceso de identificación, se dispone de los comandos siguientes:

a) **Identificación de usuario (`USER`)**

Lo primero que debe hacer un cliente POP3 es identificarse ante el servidor POP3. Uno de los métodos es hacerlo mediante el comando `USER`, dentro del cual se envía el nombre que identifica al usuario.

```
USER nombre
```

b) **Envío de la contraseña (`PASS`)**

Una vez identificado el usuario mediante el comando `USER`, debe llevarse a cabo la autenticación por medio del envío de una contraseña con el comando `PASS`. El servidor utiliza la cadena enviada en este último junto con el nombre de usuario para dar acceso al usuario al buzón o denegárselo.

```
PASS contraseña
```

c) **Identificación y autenticación del usuario con seguridad (`APOP`)**

El método de identificación y autenticación mediante los comandos `USER` y `PASS` tiene el problema de que tanto el nombre como la contraseña viajan por la red sin ningún mecanismo de seguridad. Un método alternativo de identificación y autenticación del usuario consiste en utilizar el comando `APOP`, que incorpora mecanismos de seguridad en el envío de la contraseña.

```
APOP nombre resumen
```

ANOTACIONES

Nota

El comando APOP actúa de la manera siguiente:

Al conectarse, el servidor envía una cadena al cliente. Este último concatena la cadena recibida a la contraseña y crea un resumen de la misma por medio de un algoritmo de creación de resúmenes (*hash* criptográfico). Este resumen se envía junto con el nombre identificativo del usuario como argumentos del comando APOP.

El servidor, para verificar el ususario, genera el mismo resumen y lo compara con el que ha recibido.

2) **Estado de transacción**. En este estado, el cliente solicita acciones al servidor POP3. Las acciones que puede pedir son las siguientes:

a) Estado (**STAT**)

Una vez autenticado el usuario, el cliente POP3 puede requerir información sobre el estado del buzón del usuario por medio del comando STAT que no tiene argumentos y devuelve el número de mensajes y los bytes que ocupa el buzón del usuario.

```
STAT
```

b) Listado (**LIST**)

Cuando ya se sabe el número de mensajes que hay en el buzón, el paso siguiente es la petición de información sobre uno de los mensajes o sobre todos. El POP3 proporciona el comando LIST para esta tarea. Este comando devuelve, para cada mensaje, un número de mensaje y los bytes que ocupa.

```
LIST [mensaje]
```

c) **Recuperación de mensajes (RETR)**

Una vez se conocen los mensajes que hay en el buzón, deben recuperarse los que quiere leer. Ello puede hacerlo el cliente POP3, para

ANOTACIONES

cada mensaje, con el comando RETR, que incluye como argumento el identificador del mensaje que se quiere recuperar.

```
RETR mensaje
```

d) Borrado de mensajes (**DELE**)

Tras leer un mensaje, puede interesar borrarlo. El comando DELE indica al servidor POP3 que marque como mensaje a borrar el identificado como tal en el argumento; sin embargo, el mensaje no se borrará hasta que no se entre en el estado de actualización.

```
DELE mensaje
```

e) Operación nula (**NOOP**)

El comando NOOP sirve para saber si la conexión con el servidor todavía está abierta. Con dicho comando, el servidor POP3 no hace nada, excepto devolver una respuesta afirmativa.

```
NOOP
```

f) Desmarcado de mensajes para borrar (**RSET**)

Una vez se ha llevado a cabo una llamada al comando DELE, y antes de entrar en el estado de actualización, el usuario puede echarse atrás y pedir al servidor POP3, mediante el comando RSET, que desmarque todos los mensajes marcados para borrar.

```
RSET
```

g) Recuperación de la parte superior de un mensaje (**TOP**)

En ocasiones, puede interesar recuperar sólo la parte superior de un mensaje para decidir si vale la pena recuperarlo todo o no. El comando TOP permite recuperar la cabecera y *n* líneas del mensaje identificado en el argumento.

```
TOP mensaje n
```

ANOTACIONES

h) Lista de identificadores únicos (**`UIDL`**)

Todos los mensajes del buzón tienen un identificador único (*unique-id*) permanente (a diferencia del número de mensaje, que es único dentro del buzón, pero que puede ir variando). El comando `UIDL` permite obtener el número de mensaje y el identificador único de uno de los mensajes del buzón o de todos.

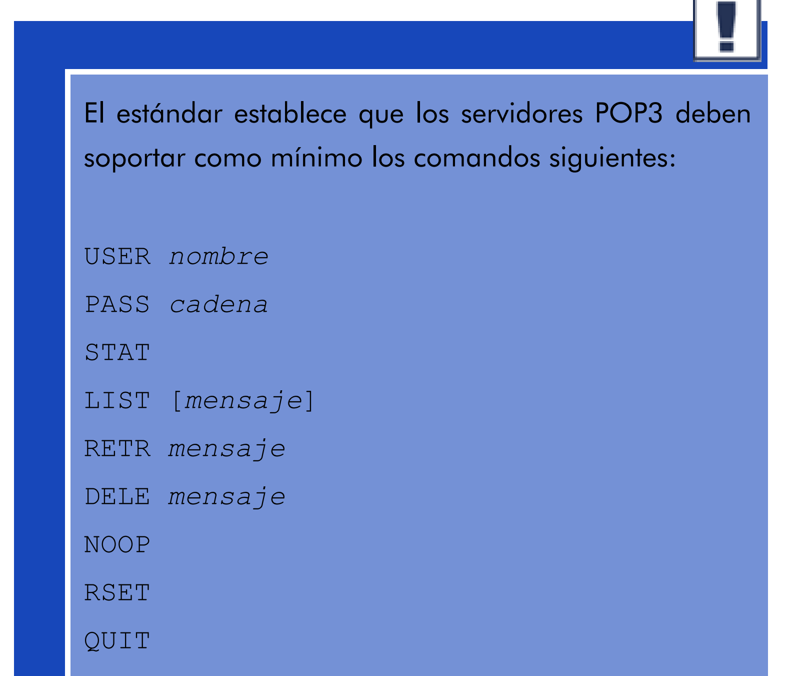
```
UIDL [mensaje]
```

i) Paso al estado de actualización (**`QUIT`**)

Una vez finalizadas las transacciones, el cliente POP3 debe llamar el comando `QUIT` para pasar al estado de actualización.

```
QUIT
```

3) **Estado de actualización.** En este estado, el servidor POP3, en primer lugar, borra todos los mensajes marcados para borrar y, con posterioridad, libera todos los recursos y cierra la conexión TCP. El estado de actualización no dispone de ningún comando asociado.



El estándar establece que los servidores POP3 deben soportar como mínimo los comandos siguientes:

```
USER nombre
PASS cadena
STAT
LIST [mensaje]
RETR mensaje
DELE mensaje
NOOP
RSET
QUIT
```

ANOTACIONES

19.3.4. Ejemplo

A continuación, se presenta un ejemplo de los comandos de acceso de un cliente POP3 (en negrita) a un servidor POP3 para recuperar el mensaje enviado en el ejemplo anterior:

```
+OK QPOP (version 2.52) at pop.uoc.es starting.
USER rmarti
+OK Password required for rmarti.
PASS prueba
-ERR Password supplied for "rmarti" is incorrect.
+OK Pop server at pop.uoc.es signing off.
PASS password
+OK rmarti has 6 message(s) (190885 bytes).
STAT
+OK 6 190885
LIST
+OK 6 messages (190885 bytes)
1 3140
2 3326
3 1911
4 180846
5 861
6 801
.
RETR 6
+OK 801
Received: from campus.uoc.es by peru.uoc.es
   (8.8.5/8.8.5) with ESMTP id SAA14826
   for <rmarti@uoc.edu>; Fri, 27 Jun 2003
   18:35:52 +0200 (MET DST)
From: Jordi Inyigo <jinyigo@uoc.edu>
Message-Id:
   <199809211639.SAA20364@peru.uoc.es>
To: rmarti@uoc.edu, xperramon@uoc.edu
Subject: Master de software libre
Date: 27 Jun 2003
Content-Type: text
Status: RO

Este es un mensaje de correo de ejemplo.
.
QUIT
+OK Pop server at dns signing off
```

ANOTACIONES

19.4. Acceso complejo a los buzones de correo: el IMAP4rev1

El protocolo de acceso a mensajes Internet, versión 4rev1, IMAP4rev1, permite al cliente acceder a los mensajes de correo electrónico de un servidor y manipularlos.

El IMAP4rev1 (a partir de ahora lo llamaremos *IMAP4*) permite al usuario disponer de diferentes buzones estructurados de manera jerárquica y, al mismo tiempo, poderlos manipular de manera remota, tal como se hace con los buzones locales.

Nota

El IMAP4rev1 surgió de la evolución de las especificaciones IMAP2 [RFC 1176], IMAP3 [RFC 1203] e IMAP4 [RFC 1730].

El IMAP4 también proporciona a los clientes la capacidad de resincronización con el servidor.

El IMAP4 no especifica ningún método para el envío de correo; otros protocolos de transferencia de correo, como el SMTP, proporcionan esta funcionalidad.

19.4.1. Modelo del IMAP4

El modelo funcional del IMAP4 se basa en los elementos que presentamos a continuación:

- **Agente de usuario**: utiliza el cliente IMAP4 para leer el correo de su buzón.
- **Cliente IMAP4**: se comunica con el servidor IMAP4 por medio del IMAP4 para acceder a su buzón de correo.
- **Servidor IMAP4**: recibe peticiones de los clientes IMAP4 y se las sirve accediendo a los buzones correspondientes.

La figura siguiente presenta los elementos del modelo funcional del IMAP4 integrados en un sistema en el que se utiliza SMTP para enviar el correo e IMAP4 para acceder a los buzones:

ANOTACIONES

Lectura complementaria

Si queréis más información sobre el IMAP4rev1, consultad la obra siguiente:

M. Crispin (1996, diciembre). *RFC 2060 - Internet Message Access Protocol - Version 4rev1.*

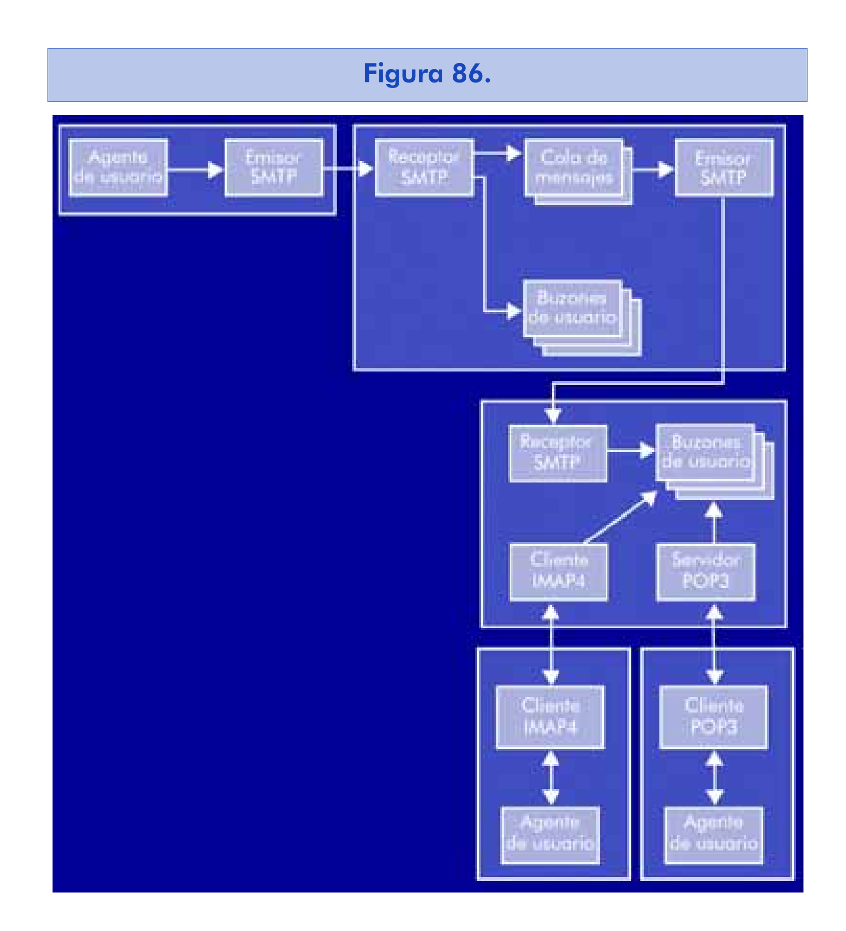


Figura 86.

19.4.2. Conceptos básicos del IMAP4

El IMAP4 puede utilizarse con cualquier protocolo de transporte fiable. Por norma general, se utiliza el TCP y, en este caso, se utiliza el puerto 143. Todas las interacciones entre cliente y servidor se llevan a cabo en forma de líneas ASCII acabadas con un carácter `<CRLF>`.

Cada comando del cliente empieza con un identificador (por lo general, una cadena alfanumérica corta) llamado ***tag***. El cliente debe generar un *tag* diferente para cada comando.

El servidor puede enviar datos tanto en respuesta a un comando del cliente, como de manera unilateral, y el cliente debe estar a punto para recibirlos en todo momento. Los datos transmitidos por el servidor hacia el cliente y las respuestas de estatus que no implican la finalización del comando empiezan con el *token*. La respuesta final de culminación de comando empieza con el mismo *tag* que el comando del cliente que ha dado lugar a la respuesta.

ANOTACIONES

Además de las respuestas específicas de cada comando, casi todos los comandos disponen, como mínimo, de los **resultados de estatus** siguientes:

```
OK [Parámetros]: el comando se ha ejecutado.
NO [Parámetros]: el comando no se ha ejecutado.
BAD [Parámetros]: comando desconocido o argumentos inválidos.
```

Nota

Consultad el comando LOGOUT en el apartado 19.4.3.

Ejemplo

```
a006 logout
* BYE IMAP4rev1 server terminating connection
a006 OK LOGOUT completed
```

La primera línea es la llamada al comando precedida del *tag* (a006). En este caso, el comando LOGOUT. Después, se puede observar la respuesta, que está formada por dos líneas. La última línea de la respuesta empieza con el mismo *tag* que la llamada, mientras que las líneas anteriores lo hacen con el *token*.

El cliente puede enviar un comando después de otro sin esperar el resultado del primero. De manera similar, un servidor puede empezar a procesar un comando antes de acabar de procesar el que se ejecuta en aquel momento.

Además del texto, cada mensaje tiene asociados diferentes atributos que se encuentran almacenados dentro del buzón y que son son los siguientes:

1) **Identificador único (UID, *unique identifier*)**: a cada mensaje se le asocia un valor de 32 bits que, cuando se utiliza conjuntamente con los valores de validez de identificador único (*unique identifier validity value*), forma un valor de 64 bits que identifica de manera única un mensaje dentro de un buzón. Los UID se asignan de manera ascendente dentro del buzón, pero no necesariamente de manera contigua.

2) **Número de secuencia del mensaje (*message sequence number*)**: este atributo proporciona la posición relativa del mensaje dentro del buzón, desde el 1 hasta al número total de mensajes.

ANOTACIONES

Esta posición debe ordenarse siguiendo los UID ascendentes. Los números de secuencia del mensaje se pueden reasignar durante la sesión (por ejemplo, cuando se elimina un mensaje del buzón).

3) **Indicadores**: cada mensaje tiene asociada una lista de cero indicadores, o más, que informan del estado:
   - `\Seen`: mensaje leído
   - `\Answered`: mensaje contestado
   - `\Flagged`: mensaje marcado por atención urgente/especial
   - `\Deleted`: mensaje marcado para ser borrado por un *Expunge* posterior
   - `\Draft`: mensaje no editado del todo
   - `\Recent`: mensaje acabado de llegar en esta sesión

4) **Fecha interna** (***internal date***): fecha y hora que cada mensaje lleva asociadas de manera interna dentro del servidor, que reflejan cuándo ha llegado el mensaje al servidor. No es la fecha del mensaje RFC 822.

5) **Longitud [RFC 822]** (**[RFC 822]** ***size***): es el número de bytes del mensaje expresado en el formato RFC 822.

6) **Estructura del sobre** (***envelope structure***): representación analizada de la información del sobre RFC 822.

7) **Estructura del cuerpo** (***body structure***): representación analizada de la información de la estructura del cuerpo MIME.

8) **Textos de mensaje**: además de permitir la recuperación de los mensajes RFC 822 enteros, con el IMAP4 también se pueden efectuar recuperaciones parciales. En concreto, permite recuperar la cabecera y/o el cuerpo del mensaje RFC 822, una parte del cuerpo MIME o una cabecera MIME.

El IMAP4 especifica cuatro estados:

- **Estado no autenticado**: el cliente debe proporcionar sus credenciales. Se llega a este estado cuando se empieza una conexión, salvo que ya se haya preautenticado.

- **Estado autenticado**: el cliente debe seleccionar un buzón al que accederá antes de tener permiso para efectuar comandos sobre los mensajes.

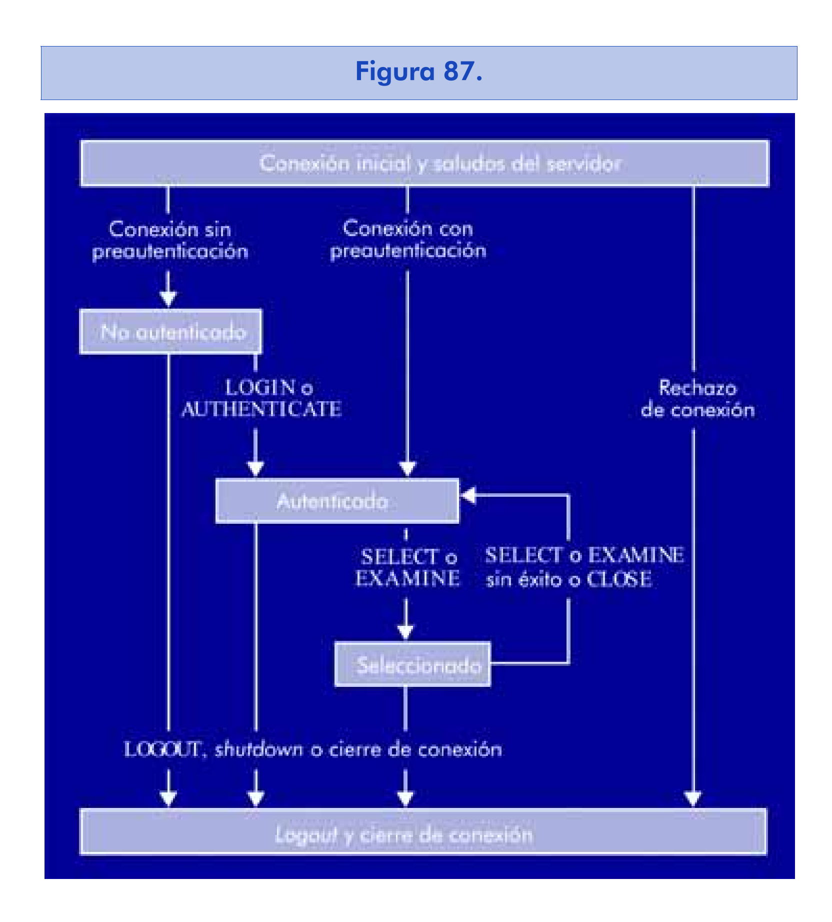
ANOTACIONES

- **Estado seleccionado**: se entra en este estado cuando se ha seleccionado con éxito un buzón.

- **Estado de *logout***: la conexión se acaba y el servidor la cerrará. Se puede entrar en este estado como resultado de una petición del cliente o de manera unilateral por parte del servidor.

La figura siguiente muestra el diagrama de flujo entre los diferentes estados definidos por el IMAP4:

Figura 87.



Cada usuario tiene su correo en el servidor, depositado en un conjunto de buzones estructurados jerárquicamente que se pueden manipular remotamente a través del IMAP4.

Nota

Aunque el IMAP4 utiliza la palabra *buzón* para identificar todos los sitios donde se pueden guardar mensajes, en realidad lo que se entiende por *buzón donde se reciben los mensajes* es la bandeja de entrada (buzón *inbox*), mientras que los otros buzones son más bien carpetas en las que se clasifican estos mensajes recibidos.

Cada buzón se identifica por un nombre relativo, una cadena de caracteres que lo diferencia de los buzones hermanos. Asimismo, cada buzón dispone de un nombre que lo identifica dentro de la estructura formada por la secuencia de los nombres relativos de los buzones que definen los niveles de jerarquía superiores, de izquierda a derecha, separados con un único carácter (por norma general, el carácter "/"). Debe utilizarse el mismo separador en todos los niveles de la jerarquía dentro de un nombre.

19.4.3. Funcionalidad del IMAP4

La funcionalidad proporcionada por el IMAP4, como la del POP3, imita la que se requiere para el acceso a los buzones de correo postal. Como mejora respecto al POP3, el IMAP4 proporciona una estructura jerárquica del buzón en forma de carpetas, así como facilidades de suscripción, lo que da lugar a nuevos comandos que permiten gestionar todos estos elementos.

Las funciones que se utilizarán según el estado en que se encuentre la comunicación serán las siguientes:

1) **Cualquier estado (comandos/estados universales)**. Sea cual sea el estado de la conexión, se pueden utilizar los comandos siguientes:

a) **Petición de capacidades (CAPABILITY)**

El comando CAPABILITY, que no tiene ningún argumento, sirve para solicitar la lista de capacidades que soporta el servidor

```
CAPABILITY
```

b) **Operación nula (NOOP)**

El comando NOOP permite al cliente IMAP4 averiguar si la conexión con el servidor todavía está abierta.

```
NOOP
```

c) **Finalización de conexión (LOGOUT)**

El comando LOGOUT permite al cliente notificar al servidor que quiere acabar la conexión.

```
LOGOUT
```

ANOTACIONES

2) **Estado no autenticado**. En este estado, un cliente proporciona sus credenciales; para ello, se utilizan los comandos siguientes:

a) **Indicador de autenticación (`AUTHENTICATE`)**

El comando `AUTHENTICATE` sirve para indicar al servidor IMAP4 un mecanismo de autenticación; es decir, cuál de los diferentes mecanismos de autenticación posibles utiliza el cliente.

```
AUTHENTICATE tipo_autenticación
```

b) **Identificación de usuario (`LOGIN`)**

Como todo protocolo, el IMAP4 proporciona un comando para permitir que el cliente se identifique ante el servidor por medio del envío de un identificador de usuario y una contraseña. Este comando es `LOGIN`.

```
LOGIN id_usuario contraseña
```

3) **Estado autenticado**. En este estado el IMAP4 proporciona algunas funcionalidades nuevas:

a) **Selección de un buzón** (`SELECT`)

Una vez autenticado, el cliente IMAP4 puede seleccionar un buzón para acceder a sus mensajes. El comando `SELECT` proporciona esta funcionalidad.

```
SELECT buzón
```

Este comando, además de los resultados de estatus, retorna las respuestas obligatorias sin *tag* siguientes:

```
FLAGS:   informa de los identificadores que se pueden
         aplicar al buzón del comando.
EXISTS: número de mensajes existentes en el buzón.
RECENT: número de mensajes con el indicador \Recent.
```

También puede retornar las respuestas `OK` sin *tag*:

```
NSEEN: número del primer mensaje sin el indicador \Seen.
PERMANENTFLAGS: lista de indicadores que pueden modificarse
permanentemente.
```

ANOTACIONES

b) Examen de un buzón (**EXAMINE**)

El comando EXAMINE permite un acceso al buzón similar al del comando SELECT, pero de manera que el buzón sea sólo de lectura. Este comando, además de los resultados de estatus, puede retornar las mismas respuestas que el comando SELECT.

```
EXAMINE buzón
```

c) Creación de un buzón (**CREATE**)

Al usuario le puede interesar crear un buzón con un nombre determinado. El IMAP4 proporciona el comando CREATE para ello.

```
CREATE buzón
```

d) Borrado de un buzón (**DELETE**)

El IMAP4 también facilita la eliminación permanente de un buzón con un nombre determinado por medio del comando DELETE.

```
DELETE buzón
```

e) Renombramiento de un buzón (**RENAME**)

En ocasiones, el usuario desea cambiar el nombre de un buzón por uno nuevo. El cliente IMAP4 puede solicitar esta acción al servidor por medio del comando RENAME.

```
RENAME buzón  nuevobuzón
```

f) Suscripción de un buzón (**SUBSCRIBE**)

En el IMAP4, no es preciso que todos los buzones estén activos a cada momento. Con el comando SUBSCRIBE, el nombre del buzón pasa a formar parte de la lista de buzones activos o suscritos.

```
SUBSCRIBE buzón
```

g) Eliminación de la suscripción de un buzón (**UNSUBSCRIBE**)

El comando UNSUBSCRIBE permite desactivar un buzón eliminando su nombre de la lista de buzones activos o suscritos.

```
UNSUBSCRIBE buzón
```

ANOTACIONES

h) **Listado de buzones** (`LIST`)

El comando LIST permite obtener los nombres de buzones que cumplen los criterios de búsqueda deseados dentro de un buzón de referencia.

```
LIST buzón 1*criterio_búsqueda
```

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

`LIST`: nombre del buzón que cumple los criterios de búsqueda deseados.

Puede haber más de una respuesta `LIST`.

i) **Listado de buzones suscritos** (`LSUB`)

El comando `LSUB` permite obtener los nombres de los buzones activos o suscritos que cumplen los criterios de búsqueda deseados dentro de un buzón de referencia.

```
LSUB buzón 1*criterio_búsqueda
```

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

`LSUB`: nombre del buzón que cumple los criterios de búsqueda deseados.

Puede haber más de una respuesta `LSUB`.

j) **Estado del buzón** (`STATUS`)

El comando `STATUS` permite conocer el estado de un buzón.

Los atributos de estado definidos son los siguientes:

- `MESSAGE`: número de mensajes en el buzón.
- `RECENT`: número de mensajes con el indicador `\Recent`.
- `UIDNEXT`: UID que se asignará al mensaje siguiente que llegue al buzón.
- `UIDVALIDITY`: valor del UID del buzón.
- `UNSEEN`: número de mensajes sin el indicador `\Seen`.

```
STATUS buzón
(1#atributo_estado)
```

ANOTACIONES

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
STATUS: nombre de buzón que cumple el estatus deseado y
        la información de estatus requerida.
```

k) Añadido de un mensaje al buzón (APPEND)

El comando `APPEND` permite al cliente el IMAP4 añadir un texto literal como nuevo mensaje al final de un buzón seleccionado, con la fecha, la hora y los indicadores deseados.

```
APPEND buzón[lista_flags]
  [fecha_hora] literal
```

4) **Estado seleccionado**. En el estado seleccionado, el IMAP4 proporciona nuevas funcionalidades, además de los comandos universales y los del estado autenticado:

a) Control del buzón (CHECK)

El comando `CHECK` permite al cliente IMAP4 pedir al servidor un punto de control del buzón seleccionado en un momento determinado.

```
CHECK
```

b) Cierre del buzón (CLOSE)

El comando `CLOSE` permite cerrar un buzón seleccionado y eliminar permanentemente todos sus mensajes que tienen el indicador `\Deleted`.

```
CLOSE
```

c) Eliminación de mensajes (EXPUNGE)

El comando `EXPUNGE` permite eliminar de manera permanente todos los mensajes que tienen el indicador `\Deleted` del buzón seleccionado y sin necesidad de cerrarlo.

```
EXPUNGE
```

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
EXPUNGE: número de secuencia de mensaje especificado que
         ha sido eliminado permanentemente.
```

ANOTACIONES

d) Búsqueda de mensaje (SEARCH)

El comando SEARCH permite al cliente IMAP4 buscar dentro del buzón los mensajes que contienen un conjunto de caracteres especificado y que cumplen unos criterios de búsqueda deseados.

```
SEARCH [CHARSET
  conjunto_caracteres]
  1#criterio_búsqueda
```

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
SEARCH: un número de secuencia o más de los mensajes que
        cumplen el criterio de búsqueda deseado.
```

e) Recuperación de mensajes (FETCH)

El comando FETCH permite la recuperación de un conjunto de mensajes (total o parcialmente), especificando unos atributos de recuperación.

```
FETCH conjunto_mensajes ALL | FULL
  | FAST |
  atrib_recup | (1#atrib_recup)
```

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
FETCH: información del mensaje que presenta los atributos
       de recuperación especificados.
```

f) Modificación de almacén (STORE)

El comando STORE permite modificar los indicadores del almacén que proporcionan los atributos a un conjunto de mensajes del buzón.

```
STORE conjunto_mensajes
  indicadores_atrib_almacén
```

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
FETCH: información de los mensajes.
```

g) Copia de mensaje(s) (COPY)

El comando COPY permite copiar un conjunto de mensajes al final de un buzón especificado.

```
COPY conjunto_mensajes buzón
```

h) Retorno de identificador único (`UID`)

La sentencia `UID`, seguida de los parámetros `COPY`, `FETCH`, `STORE` o `SEARCH`, en lugar de retornar el número o números de secuencia de mensaje, retorna sus identificadores únicos.

```
UID (COPY ... | FETCH ... |
  SEARCH ... | STORE ...)
```

Este comando, además de los resultados de estatus, puede retornar las respuestas sin *tag* siguientes:

```
FETCH: información del mensaje que presenta los atributos
       de recuperación especificados.
SEARCH: un UID o más indican los mensajes que cumplen el
        criterio de búsqueda deseado.
```

5) Experimental/expansión

Comando experimental (`X<atom>`)

El IMAP4 permite especificar comandos experimentales que no son una parte de la especificación, siempre que su nombre empiece con el prefijo `X`.

```
X1*carácter
```

19.4.4. Ejemplo

A continuación, se presenta un ejemplo de los comandos de un cliente IMAP4 (en negrita) que accede a un servidor IMAP4 para entrar en sus buzones. En el ejemplo se ve cómo se recupera la información del mensaje 12 (`fetch 12 full`). En la información retornada por el servidor, se ve la información "parseada" de la cabecera. A continuación, se recupera toda la cabecera del mismo mensaje (`fetch12 body [header]`). Al final, antes de cerrar la conexión, se marca el mensaje para que sea borrado.

```
* OK IMAP4rev1 Service Ready
a001 login rmarti secret
a001 OK LOGIN completed
a002 select inbox
* 18 EXISTS
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* 2 RECENT
```

ANOTACIONES

```
* OK [UNSEEN 17] Message 17 is the first unseen message
* OK [UIDVALIDITY 3857529045] UIDs valid
a002 OK [READ-WRITE] SELECT completed
a003 fetch 12 full
* 12 FETCH (FLAGS (\Seen) INTERNALDATE "27-Jun-2003
02:44:25 -0700" RFC 822.SIZE 4286 ENVELOPE
("Fri, 27 Jun 2003 02:23:25 -0700 (PDT)"
"Ejemplo de acceso IMAP4rev1"
(("Jordi Inyigo" NIL "jinyigo" "uoc.edu"))
(("Jordi Inyigo" NIL "jinyigo" "uoc.edu"))
(("Jordi Inyigo" NIL "jinyigo"
((NIL NIL "rmarti" "uoc.edu"))
((NIL NIL "xperramon" "uoc.edu")
("JM Marques" NIL "jmmarques" "uoc.edu"))
NIL NIL
“<B27397-0100000@uoc.edu>")
BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL
   "7BIT" 3028 92))
a003 OK FETCH completed
a004 fetch 12 body[header]
* 12 FETCH (BODY[HEADER] {350}
Date: Fri, 27 Jun 2003 02:23:25 -0700 (PDT)
From: Jordi Inyigo <jinyigo@uoc.edu>
Subject: Ejemplo de acceso IMAP4rev1
To: rmarti@uoc.edu
cc: xperramon@uoc.edu,
JM Marques <jmmarques@uoc.edu>
Message-Id: <B27397-0100000@uoc.edu>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; CHARSET=US-ASCII

)
a004 OK FETCH completed
a005 store 12 +flags \deleted
* 12 FETCH (FLAGS (\Seen \Deleted))
a005 OK +FLAGS completed
a006 logout
* BYE IMAP4rev1 server terminating connection
a006 OK LOGOUT completed
```

19.5. Extensiones multimedia: el formato MIME

La norma RFC 822 define un formato de mensaje y un contenido con una única parte de texto en ASCII de 7 bits. Se vio que este formato era muy pobre y que se precisaba algún método para superar sus limitaciones.

El formato **MIME** (*multipurpose Internet mail extensions*) redefine el formato del mensaje para permitir, sin perder la compatibilidad con el formato definido por el RFC 822, las características siguientes:

- Contenido de texto no sólo ASCII de 7 bits.
- Contenido no texto.
- Contenido con múltiples partes.
- Cabeceras con texto no sólo ASCII de 7 bits.

19.5.1. Nuevos campos de cabecera

Para permitir todas las nuevas extensiones, el MIME define nuevos campos de cabecera: `MIME-Version`, `Content-Type`, `Content-Transfer-Encoding`, `Content-ID` y `Content-Description`.

Indicador de versión (`MIME-Version`)

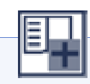
El campo de cabecera `MIME-Version` indica la versión MIME que se utiliza en el mensaje (la versión de MIME que se utiliza actualmente, y que especificamos en este apartado, es la 1.0.). Este campo es útil para que el receptor del mensaje pueda interpretar los campos de cabecera MIME.

```
MIME-Version: 1*digit.1*digit
```

Indicador del tipo y subtipo de datos (`Content-Type`)

El campo de cabecera `Content-Type` permite indicar los tipos y subtipos de los datos que se encuentran dentro del mensaje. De este modo, el receptor podrá conocer los tipos de datos que contiene el mensaje y podrá visualizarlos adecuadamente. Según el tipo o subtipo de dato, puede incluir algún parámetro adicional.

```
Content-Type: tipo/subtipo *(; parámetro)
tipo = tipo-discreto | tipo-compuesto
tipo-discreto = text | image | audio | video |
application | extension-token
tipo-compuesto = message | multipart |
extension-token
```



Para obtener más información sobre el formato de mensajes MIME, podéis consultar las obras siguientes:

N. Freed; N. Borenstein (1996, noviembre). *RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.*

ANOTACIONES

N. Freed; N. Borenstein (1996, noviembre). *RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.*

K. Moore (1996, noviembre). *RFC 2047 - MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header
Extensions for Non-ASCII Text.*

N. Freed; J. Klensin; J. Postel (1996, noviembre). *RFC 2048 - Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures.*

N. Freed; N. Borenstein (1996, noviembre). *RFC 2049 - Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples.*

La norma define cinco tipos de contenido (`tipo-discreto`), con los subtipos correspondientes. Estos tipos corresponden a datos monomedia:

- `text`: para información de texto.
- `image`: para imágenes.
- `audio`: para sonido.
- `video`: para vídeo.
- `application`: para cualquier otro tipo de datos, por norma general en formato específico de alguna aplicación.

La norma también define dos tipos de datos compuestos (`tipo-compuesto`):

- `multipart`: para datos formados de muchas partes o para datos independientes dentro de un mismo mensaje.

Nota

En los mensajes de tipo `multipart`, el parámetro *boundary* incluye una cadena, precedida por dos caracteres de guión, "--", que se utiliza como separador entre las diferentes partes del mensaje.

ANOTACIONES

Para indicar que se han acabado todas las partes, se utiliza la cadena del parámetro *boundary*, precedida y seguida de dos caracteres de guión, "--".

- `message`: para encapsular otro mensaje dentro del mensaje.

La tabla siguiente presenta los valores de tipo, subtipo y parámetros para los `Content-Type` definidos en la norma:

Tabla 13.

Valores definidos para `Content-Type`

Tipo	Subtipo	Parámetros
text	plain	charset = ISO-8859-(1 | ... | 9) | us-ASCII
	enriched	charset = ISO-8859-(1 | ... | 9) | us-ASCII
image	gif	–
	jpeg	–
audio	basic	–
video	mpeg	quicktime
application	octet-stream	type = *cadena*; padding = *entero*
	postscript	–
multipart	mixed	boundary = *cadena*
	alternative	boundary = *cadena*
	parallel	boundary = *cadena*
	digest	boundary = *cadena*
message	rfc 822	–
	partial	id = *cadena*; number = *entero* [;total = *entero*]
	external-body	access-type = ftp | anon-ftp | tftp | afs | local-file | mail-server [;expiration = *date-time*] [;size = *entero*] [;permission=read | read-write] [;name = *cadena*] [;site = *cadena*] [;dir = *cadena*] [;mode = netascii | octet | mail | ascii | ebcdic | image | localn] [;server = *cadena*] [;subject = *cadena*]

ANOTACIONES

Especificador del tipo de codificación (`Content-Transfer-Encoding`)

En algunos protocolos, como el SMTP, la información que se envía debe ser en ASCII de 7 bits. En ocasiones, los datos originales no tendrán este formato y, entonces, será preciso aplicarles algún tipo de codificación antes de enviarlos.

El campo de cabecera `Content-Transfer-Encoding` sirve para especificar el tipo de codificación que se ha aplicado al contenido del mensaje para que el receptor pueda descodificarlo, si es preciso. La norma define diferentes tipos de codificación:

```
Content-Transfer-Encoding: mecanismo
mecanismo = 7bit | 8bit | binary | quoted-printable | base64
```

a) Codificación `7bit | 8bit | binary`

Los mecanismos de codificación `7bit`, `8bit` y `binary` sólo sirven para indicar de qué tipo son los datos transmitidos. En estos casos, los datos no se codifican y, por norma general, se utilizan para aplicaciones que no restringen la representación de datos en 8 bits.

b) Codificación `quoted-printable`

El mecanismo de codificación `quoted-printable` se utiliza para la representación y codificación en ASCII de 7 bits de datos, la mayoría de los cuales ya es de bytes representables en este formato. Es decir, este mecanismo se aplica cuando la información es mayoritariamente de caracteres de texto. Las normas básicas de codificación son las siguientes:

- Cualquier byte se puede representar con el carácter "=" seguido por la notación hexadecimal en dos dígitos (y letras en mayúscula) del valor del byte.
- Los bytes con valores decimales entre 33-60 y 62-126, incluídos los cuatro, se pueden representar con el carácter ASCII correspondiente.

c) Codificación `Base64`

El mecanismo de codificación `Base64` ofrece una codificación de información binaria en ASCII de 7 bits que no deba ser legible. Los algoritmos de codificación y descodificación son muy simples.

El proceso de codificación se lleva a cabo tomando grupos de 3 bytes (24 bits). Manteniendo el orden de bits original, estos 24 bits se reagrupan en 4 bloques de 6 bits (6 bits = 64 combinaciones).

El fichero codificado se obtiene tomando cada uno de estos bloques de 6 bits y codificándolo como un carácter alfanumérico a partir del valor binario de los 6 bits, según la tabla siguiente:

Nota

La codificación de datos en `Base64` utiliza 4 caracteres para cada 3 bytes. Por este motivo, `Base64` aumenta el tamaño de la información un 33% (4/3).

Tabla 14.

Tabla de codificación `Base64`

Valor	Carácter	Valor	Carácter	Valor	Carácter	Valor	Carácter
0	'A'	26	'a'	52	'0'	62	'+'
1	'B'					63	'/'
...		...		...		pad	'='
25	'Z'	51	'z'	61	'9'		

Nota

Si debe codificarse un número de bytes que no sea múltiplo de 3 (la codificación, por tanto, no sería múltiplo de 4 bytes), se codifican todos los bytes y, al final de la cadena de caracteres ya codificada, se añaden tantos caracteres "=" (carácter para rellenar, *pad*) como sean necesarios (máximo dos) hasta llegar a un número de caracteres múltiplo de 4.

En el proceso de descodificación, se toman los caracteres alfanuméricos recibidos y se reconvierten en su valor binario correspondiente (6 bits) según la tabla. Cada 4 caracteres recibidos dan lugar a 24 bits, y sólo es preciso irlos reagrupando en 3 bytes para generar el fichero descodificado. Antes de descodificar todo el mensaje, es preciso eliminar todos los caracteres "=" que se encuentren al final.

Identificador del contenido del mensaje (`Content-ID`)

El campo de cabecera `Content-ID` se utiliza para proporcionar un identificador único al contenido del mensaje. Con dicho identificador, se hace referencia al contenido de manera no ambigua.

```
Content-ID: id-msg
```

ANOTACIONES

Informador descriptivo del contenido (`Content-Description`)

El campo `Content-Description` proporciona información descriptiva del contenido en forma de texto.

```
Content-Description: texto
```

19.5.2. Extensiones para texto no ASCII en las cabeceras

El MIME también permite codificar texto no ASCII en las cabeceras de los mensajes RFC 822 de manera no ambigua. Este método permite codificar toda la cabecera o sólo una parte. El texto (que puede ser de más de un carácter) ya codificado, precedido por el conjunto de caracteres y un identificador del tipo de codificación que se ha aplicado, se incluye en el lugar correspondiente de la cabecera entre los caracteres "=?" y "?=":

```
=? charset ? codificación ? texto-codificado ?=
charset = ISO-8859-(1 | ... | 9) | us-ASCII
codificación = Q | B
```

Nota

a) **Conjunto de caracteres** (*charset*): es válido cualquiera de los caracteres permitidos en el `Content-Type text/plain`.

b) **Codificaciones**: el formato MIME permite dos tipos de codificación similares a las codificaciones que se pueden aplicar al cuerpo del mensaje:

- `Q`: es parecida al `Content-Transfer-Encoding quoted-printable`. Es la que se recomienda cuando la mayoría de los caracteres que deben codificarse son ASCII.
- `B`: es idéntica al `Content-Transfer-Encoding Base64`. Es adecuada para el resto de casos.

ANOTACIONES

19.5.3. Mensajes multiparte

En los mensajes multiparte, cada una de las partes suele estar formada por una pequeña cabecera y el contenido. En la cabecera se pueden encontrar los campos Content-Type, Content-Transfer-Encoding, Content-ID y Content-Description. Todos se refieren al contenido de la parte en cuestión.

19.5.4. Ejemplo

En este ejemplo se puede contemplar un mensaje RFC 822 con extensiones MIME versión 1.0. Es un mensaje de dos partes: la primera es de texto codificado con quoted-printable y la segunda es una imagen en formato .gif codificada en Base64. Asimismo, se puede ver un campo de cabecera con una letra *í* codificada con codificación Q.

```
Date: 27 Jun 2003 0932 PDT
From: Jordi Inyigo <jinyigo@uoc.edu>
To: Ramon =?iso-8859-1?Q?Mart=ED?=
  <rmarti@uoc.edu>
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="=_250699_"

--=_250699_
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Esto es un mensaje RFC 822 que contiene MIME.

--=_250699_
Content-Type: image/gif; name="Readme.gif"
Content-Transfer-Encoding: base64

R0lGODlhIAAgAIAAAAAAAP///yH5BAEAAAAALAAAAAAgACAAA
AJzjI+pywoQXoSywoaontzeRhnXKJYc82GOuaoL2brJlx7pg+
eSzc9yHAHqhpmi8ddLspCT3Y2pXNZ+UxpUpGNNZVsYdBsMf4l
BsMocuqLFyCHO5I6/MWY2sy7PL4F3+nU/kxcHePTl91dnR+WS
yAcimGVQAAA7

--=_250699_--
```

Nota

La "í" de *Ramon Martí* se ha codificado con codificación Q

ANOTACIONES

20. Servicio de noticias: el NNTP

El servicio de noticias (en inglés, *news*) permite el envío de mensajes, como el servicio de correo electrónico, pero con la diferencia de que el originador no especifica el destinatario o destinatarios, sino que cualquier usuario con acceso al servicio puede leerlos. Esta funcionalidad, pues, se puede comparar a la de un tablón de anuncios, en el que todo el mundo puede leer los mensajes que se encuentran colgados.

El servicio de noticias en Internet se conoce con el nombre *Usenet*, que proviene del de la red en que se desarrolló originariamente este servicio. En la actualidad, la distribución de noticias se ha extendido por todo Internet, de manera que los anuncios enviados se pueden leer en cualquier parte del mundo. En la práctica, sin embargo, es posible restringir el ámbito en que se quiere distribuir los mensajes, puesto que algunos sólo serán interesantes, por ejemplo, en una determinada zona geográfica.

20.1. El modelo NNTP

En el servicio de noticias, la distribución de los mensajes o anuncios, que en Usenet se denominan ***artículos***, se efectúa de manera descentralizada. Teniendo en cuenta el enorme volumen de tráfico que genera este servicio, no sería viable tener un servidor central en el que todo el mundo dejara sus artículos y fuera a leer los de los demás. Por este motivo, se utiliza un mecanismo de propagación en el que el autor de un artículo lo envía a un servidor de noticias, el cual se encargará de reenviarlo a una serie de servidores próximos o con los que esté conectado directamente, que a su vez lo reenviarán a otros servidores, y así sucesivamente.

Con este mecanismo de propagación, se consigue que un artículo esté disponible para ser leído, idealmente, en todos los servidores de

ANOTACIONES

la red. Los usuarios que deseen leer los artículos podrán hacerlo, pues, conectándose a cualquier servidor, preferiblemente al que tengan más próximo.

Por otro lado, los servidores sólo almacenan cada artículo durante cierto tiempo. Hay artículos que tienen una fecha de caducidad predeterminada, y el servidor los borra automáticamente en la misma. Otros se borran, por ejemplo, cuando el servidor decide que los usuarios que puedan estar interesados en los mismos ya han tenido bastante tiempo para leerlos.

El método utilizado para propagar los artículos tiene ciertas limitaciones; por ejemplo, no se puede garantizar que un artículo llegue a todos los servidores en un plazo determinado (o, en ocasiones, que llegue a un servidor concreto), es preciso controlar el camino por donde pasa cada artículo para evitar bucles, etc. Sin embargo, este método es más práctico y eficiente que la solución de un solo servidor central.

Lectura complementaria

Si queréis más información sobre el NNTP, consultad la obra siguiente:

B. Kantor; P. Lapsley (1986, febrero). *RFC 977 - Network News Transfer Protocol*.

Los servidores de noticias se comunican entre sí para intercambiarse artículos por medio del **NNTP** (*network news transfer protocol*), especificado en el documento RFC 977.

Nota

Un usuario que desee acceder al servicio, tanto para enviar artículos como para leerlos, puede ser que tenga acceso directo a alguno de los servidores de noticias. En este caso, la comunicación con el servidor es un asunto local. El caso más general, sin embargo, es que el usuario quiera acceder al servicio desde otro sistema, caso en que dicho sistema actuará como cliente. Entonces, la comunicación entre el cliente y el servidor se lleva a cabo también por medio del NNTP.

Nota

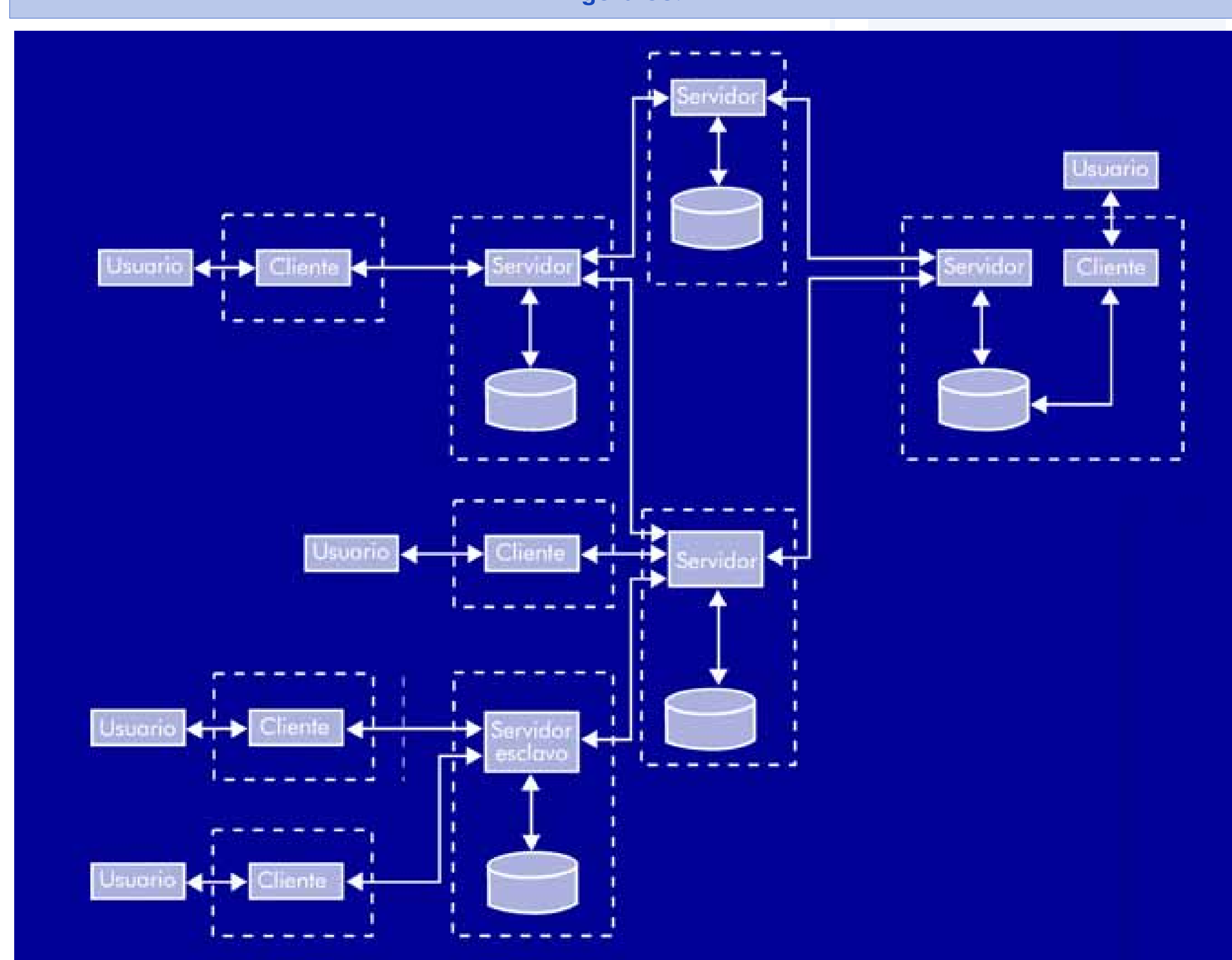
Antiguamente se utilizaban otros métodos de transmisión de artículos entre servidores, como el UUCP (*Unix to Unix copy protocol*).

Tanto si se trata de un cliente local, como de un cliente remoto, por norma general habrá otro proceso encargado de la interfaz con el usuario. Este tipo de programa se suele llamar **lector de noticias**.

ANOTACIONES

Asimismo, existe la posibilidad de que un grupo de clientes acceda a un servidor central de una organización por medio de un servidor esclavo:

Figura 88.



Nota

El servidor esclavo puede mejorar la eficiencia en el acceso, por ejemplo, almacenando localmente copias de los últimos artículos leídos, por si los solicitan otros clientes.

Los artículos disponibles en un servidor deben estar organizados en grupos, según su tema, para facilitar el acceso a los usuarios que quieran leerlos. Esta organización sigue un modelo jerárquico: en el nivel más alto de los grupos se encuentran los temas generales que,

en un segundo nivel, están divididos en subtemas, los cuales, a su vez, pueden estar subdivididos en niveles inferiores.

Nota

Es posible que un mismo artículo pertenezca a más de un grupo. Cuando un usuario envía un artículo al sistema de noticias, además de poder especificar en qué ámbito se debe distribuir, es preciso que indique obligatoriamente a qué grupo o grupos lo quiere enviar.

La nomenclatura que se utiliza para designar los grupos consiste en concatenar los nombres de cada nivel, de más alto a más bajo, separándolos con "." (por ejemplo, `news.announce` o `comp.os.linux.hardware`). En el nivel más alto de la jerarquía, los grupos originales de Usenet eran los siguientes:

- `comp`: temas relacionados con ordenadores e informática.
- `news`: artículos sobre el sistema de noticias mismo.
- `rec`: actividades recreativas, *hobbies*, etc.
- `soc`: temas sociales, culturales, humanísticos, etc.
- `sci`: temas científicos.
- `talk`: debates, discusiones, opiniones, etc.
- `misc`: otros temas no clasificables en los apartados anteriores.
- `alt`: la jerarquía alternativa, que algunos aprovechan para saltarse las normas establecidas en los grupos "oficiales".

Nota

Actualmente, existen otros grupos del nivel superior, como los que sólo contienen artículos destinados a una región geográfica. Por ejemplo `eu` para Europa, `es` para España, `fr` para Francia, etc.

En la comunidad Usenet se han establecido una serie de reglas, basadas en un sistema de votaciones, para decidir la creación de nuevos grupos o la eliminación de alguno de los ya existentes. Este carácter de democracia asamblearia, y otros aspectos como la posibilidad de publicar escritos (fotografías, vídeos, etc.) o dar a conocer las ideas de una persona de manera global y casi instantánea (un

sueño que, hasta la llegada de Internet, sólo estaba al alcance de los grandes medios de comunicación), hicieron de Usenet el gran fenómeno sociológico de Internet durante unos cuantos años.

Hoy día el servicio Usenet está eclipsado por la enorme popularidad del servicio WWW, que tiene una orientación mucho más comercial, pero no es tan participativo.

20.2. Conceptos básicos del NNTP

Por norma general, el NNTP utiliza el protocolo de transporte TCP. El número de puerto asignado al servicio de noticias es el 119.

En el NNTP, se utiliza un esquema de peticiones y respuestas como el del SMTP. Cada **petición** constituye una línea acabada con `<CRLF>` que contiene un comando, posiblemente con parámetros.

Nota

La especificación NNTP establece que las líneas de comandos no deben tener más de 512 caracteres. Por otro lado, los comandos y los parámetros se pueden escribir indistintamente en mayúsculas o minúsculas.

Las respuestas también siguen la estructura general utilizada en el FTP o el SMTP. Cada **respuesta** se representa con una línea que empieza con un código numérico de tres dígitos. A continuación, según el código de respuesta, puede haber una serie de parámetros seguidos de un texto arbitrario opcional, hasta el final de la línea, que acaba con `<CRLF>`.

Los significados del primer dígito del código de respuesta son similares a los del FTP y el SMTP; los del segundo son los siguientes:

Nota

Véase el anexo 4.

- x0x: respuesta referente a la conexión, inicialización, etc.
- x1x: selección de un grupo de noticias.
- x2x: selección de un artículo.
- x3x: distribución de artículos.
- x4x: envío de artículos.
- x8x: extensiones no estándar.
- x9x: mensajes informativos de prueba (*debugging*).

Algunas respuestas van seguidas de un texto formado por una secuencia de líneas. En este caso, cada línea acaba con `<CRLF>`, y el final de la secuencia se indica con una línea en la que sólo se encuentra el carácter "`.`" antes de `<CRLF>`. Si alguna de las lí-

ANOTACIONES

neas de la respuesta debe empezar con ".", el emisor inserta otro "." al principio de la línea y, por consiguiente, el receptor debe eliminar todos los "." iniciales que encuentre antes del final de la respuesta.

Cuando el cliente establece la conexión, el servidor responde con un código 200 ó 201 para indicar que permite que el cliente envíe artículos o que no lo permite, respectivamente.

20.3. Formato de los artículos

Lectura complementaria

Si queréis más información sobre el formato de los artículos Usenet, consultad la obra siguiente:

M.R. Horton; R. Adams (1987, diciembre). *RFC 1036 - Standard for interchange of USENET messages.*

El formato de los artículos Usenet está definido en la especificación RFC 1036, y se basa en el de los mensajes de correo electrónico (definido en la especificación 822), aunque con algunas restricciones adicionales. Por tanto, cada artículo consta de una cabecera formada por una serie de campos, y de un cuerpo separado de la misma por una línea en blanco.

Hay seis **campos obligatorios en la cabecera**, que son los que se exponen a continuación:

- `From`: dirección de correo electrónico del originador del artículo (y, opcionalmente, también su nombre).
- `Date`: día y hora en que se ha originado el artículo.
- `Newsgroups`: lista de los grupos a los que se envía el artículo, separados por comas si hay más de uno.
- `Subject`: asunto del que trata el artículo, que se utilizará como título.
- `Message-ID`: identificador único del artículo. Aparte de este identificador único, para facilitar el acceso a los artículos, cada servidor les asigna un identificador local consistente en el nombre del grupo y un número correlativo dentro del grupo. Por lo tanto, si un artículo se ha enviado a diferentes grupos, tendrá más de un identificador. Dichos identificadores sólo son significativos para un servidor, puesto que otro servidor probablemente asignará identificadores locales diferentes a los mismos artículos.

ANOTACIONES

- Path: lista de servidores por los que ha pasado el artículo, separados con símbolos de puntuación que no sean "." (por norma general, se utiliza el carácter "!"). Cada servidor debe añadir su nombre al principio de la lista. El valor de este campo permite que un servidor sepa si el artículo ya ha pasado por otro servidor y, por tanto, no es preciso volvérselo a enviar.

Asimismo, la cabecera de un artículo puede incluir los **campos opcionales** siguientes:

- Reply-To: dirección a la que deben enviarse los mensajes de correo en respuesta al artículo (por defecto, la misma del campo From).
- Sender: identificación del usuario que ha enviado el artículo al sistema de noticias, si no coincide con el del campo From.
- Followup-To: lista de grupos de noticias, separados por comas, a los que deben enviarse los artículos de respuesta (por defecto, los mismos del campo Newsgroups).
- Expires: fecha y hora en que el artículo se puede considerar caducado y, por tanto, puede ser borrado por los servidores (en ausencia de este campo, el servidor decide cuándo los borrará).
- References: lista de identificadores de otros artículos a los que se hace referencia en el artículo.
- Control: sirve para indicar que el artículo contiene un mensaje de control del sistema de noticias. El mensaje se especifica en el valor de este campo y sólo es interesante para los servidores (los usuarios no necesitan leer estos artículos). Los mensajes de control sirven para cancelar artículos, crear y borrar grupos, informar de qué artículos tiene un servidor y cuáles pide otro, etc.
- Distribution: lista de distribuciones a la que se debe enviar el artículo. Cada distribución es un conjunto de servidores que, por norma general, pertenecen a una misma área geográfica, una misma organización, etc.
- Organization: nombre de la organización (empresa, institución, etc.) a la que pertenece el originador del artículo.
- Keywords: lista de palabras clave relacionadas con el contenido del artículo.

ANOTACIONES

ANOTACIONES

- `Summary`: breve resumen del contenido del artículo.
- `Approved`: algunos grupos de noticias son moderados, lo que significa que los usuarios no pueden enviar los artículos directamente. En este caso, los artículos deben enviarse por correo electrónico a una persona, llamada **moderador del grupo**, que es la única que está autorizada para poner artículos en el grupo. De la totalidad de mensajes que recibe, el moderador decide cuáles envía al grupo y cuáles no. Los mensajes que finalmente se envían deben llevar en el campo `Approved` la dirección de correo del moderador.
- `Lines`: número de líneas del cuerpo del artículo.
- `Xref`: este campo lo genera cada servidor y no debe transmitirse de un servidor a otro. Contiene el nombre del servidor y una lista de identificadores locales que tiene el mismo artículo en otros grupos a los que se haya enviado. Una vez que el usuario ha leído el artículo, este campo permite a los lectores de noticias marcarlo como leído en todos los grupos en que aparezca.

La especificación RFC 1036 permite que la cabecera incluya otros campos no estándar.

Ejemplo

Ejemplo de artículo Usenet

```
From: usuari@acme.com (Ernest Udiant)
Path: News.uoc.es!news.rediris.es!
   news.eu.net!newsfeed.omninet.
   org!nntp.acme.com!usuari
Newsgroups: soc.culture.espanol
Subject: Nuevo libro de recetas tradi-
   cionales
Message-ID: <KvjsSAam9Ly@acme.com>
Date: Thu, 12 Dec 2002 09:12:30 GMT
Expires: Sat, 19 Dec 2002 00:00:00 GMT
Organization: ACME Inc.
Lines: 2

La semana que viene se publicará un nuevo
libro de recetas de cocina tradicional.
Seguiremos informando...
```

20.4. Comandos del NNTP

A continuación, se detallan los comandos definidos en la especificación RFC 977 del NNTP:

Nota

Consultad la tabla de los códigos de respuesta al final de este subapartado.

1) Listar grupos (LIST)

Este comando sirve para obtener una lista de los grupos de noticias disponibles. El servidor responde con un código 215 y, a continuación, envía una secuencia de líneas (acabada con "."), una para cada grupo de noticias disponible, cada una con el formato siguiente:

```
grupo último primero permiso
```

Nota

- `grupo`: nombre del grupo.
- `último`: número correlativo (identificador local) del último artículo que hay en este grupo.
- `primero`: número del primer artículo.
- `permiso`: puede ser `y` o `n` para indicar si se pueden enviar artículos a este grupo, o no, respectivamente.

```
LIST
```

2) Seleccionar un grupo (GROUP)

Este comando permite seleccionar un grupo concreto. El servidor envía una respuesta 211 con los parámetros siguientes (por este orden): número estimado de artículos del grupo, número del primer artículo, número del último y nombre del grupo. Si el grupo no existe, la respuesta es 411.

Nota

El número de artículos que hay en el grupo no tiene por qué coincidir con la diferencia entre el número del último artículo y el anterior al primero, puesto que puede ser que se hayan borrado artículos intermedios.

```
GROUP grup
```

ANOTACIONES

3) Leer el artículo **(ARTICLE)**

El parámetro puede ser un identificador único de artículo o un número de artículo dentro del grupo actualmente seleccionado (el artículo leído pasa a ser considerado como el artículo actual). Sin parámetro, este comando lee el artículo actual. El servidor responde con un código 220 seguido de dos parámetros: el número del artículo y su identificador único. A continuación, envía el contenido del artículo (cabecera y cuerpo), acabado en "`.`". Si hay algún error, el servidor responde con los códigos 412, 420, 423 o 430, según el caso.

```
ARTICLE [ < identificador > |
  número]
```

4) Leer la cabecera **(HEAD)**

Este comando es idéntico a `ARTICLE`; sin embargo, el servidor sólo devuelve la cabecera del artículo. El código de respuesta será el 221 en lugar del 220, con los mismos parámetros.

```
HEAD [ < identificador > |
  número]
```

5) Leer el cuerpo **(BODY)**

Este comando es idéntico a `ARTICLE`; sin embargo, el servidor sólo devuelve el cuerpo del artículo. El código de respuesta será el 222 en lugar del 220, con los mismos parámetros.

```
BODY [ < identificador > |
  número]
```

6) Obtener el estatus **(STAT)**

Este comando es idéntico a `ARTICLE`; sin embargo, el servidor no devuelve ningún texto, sino sólo la línea de respuesta. El código será el 223 en lugar del 220, con los mismos parámetros.



```
STAT [ < identificador > |
  número]
```

7) Seleccionar el artículo siguiente **(NEXT)**

Este comando selecciona el artículo siguiente del grupo y hace que pase a ser considerado el artículo actual (es decir, lo que se leerá si

se envía el comando llamado ARTICLE sin parámetro). El código de respuesta normal será el 223, como en el comando STAT. Un código de error específico es el 421.

```
NEXT
```

8) Seleccionar el artículo anterior (LAST)

Este comando selecciona el artículo anterior al actual. Los códigos de respuesta son como los del comando NEXT, pero cambiando el 421 por el 422.

```
LAST
```

9) Listar grupos nuevos (NEWGROUPS)

NEWGROUPS dispone de las mismas funciones que el comando LIST, excepto que el código de respuesta es 231 (en lugar de 215) y la lista de grupos está restringida a los que se han creado después de la fecha indicada por los parámetros (el primero son seis dígitos que representan año, mes y día, y el segundo, seis más que representan hora, minutos y segundos). Opcionalmente, se puede restringir todavía más la lista especificando una o más jerarquías de alto nivel separadas por comas (por ejemplo, news, rec, etc.).

```
NEWGROUPS aammdd hhmmss
  [GMT][< jerarquias >]
```

10) Listar artículos nuevos (NEWNEWS)

El servidor responde a este comando con un código 230 y, a continuación, envía una secuencia de líneas (acabada con “.”), una para cada artículo de los grupos especificados en el primer parámetro que se haya enviado o recibido después de la fecha indicada por el segundo y el tercero. Las líneas contienen los identificadores únicos de los artículos.

```
NEWNEWS grups aammdd hhmmss
 [GMT][< jerarquias >]
```

El primer parámetro es un nombre de grupo o una lista de nombres separados por comas. En un nombre, puede aparecer el carácter es-

pecial "*". En este caso, se considera que equivale a todos los nombres de grupos que, en lugar del "*", tengan una secuencia cualquiera de caracteres, que puede incluir el separador "." (Por ejemplo, `alt.biz*` puede equivaler a `alt.biz.misc`, `alt.bizarre`, etc.). Por otro lado, si un nombre tiene el prefijo "!", el grupo o grupos correspondientes se excluirán de la lista (por ejemplo, `comp.lang.*, !comp.lang.c.*` equivale a los grupos que empiecen por `comp.lang.`, pero no por `comp.lang.c.`).

Opcionalmente, se puede restringir la lista de artículos a un conjunto de una o más jerarquías si se especifican en el último parámetro separadas por comas. La lista sólo contendrá artículos que pertenezcan al menos a un grupo de las jerarquías.

11) Enviar un artículo (POST)

Este comando se utiliza para enviar un artículo. El servidor responde con un código 440 para indicar que no se permite la operación, o con un 340 para solicitar el artículo. En el segundo caso, el cliente debe enviar el contenido del artículo (cabecera y cuerpo) con el formato definido en la especificación RFC 1036 y acabado con una línea en que sólo haya un ".". Entonces, el servidor enviará la respuesta definitiva, que será 240 o, si ha habido algún error, 441.

```
POST
```

12) Ofrecer un artículo (IHAVE)

Cuando un servidor se conecta a otro, puede utilizar el comando `IHAVE` para hacerle saber de qué artículos dispone.

```
IHAVE < identificador >
```

En cada artículo, el servidor receptor enviará en la respuesta un código 335 si quiere recibir una copia del mismo o 435 si no le interesa. Si es preciso pasar el artículo, se hace por medio del comando `POST`, y el servidor receptor enviará en la respuesta un código 235 o, en caso de error, uno 436. Asimismo, puede suceder que al receptor no le interese el artículo después de examinar su contenido; en este caso, la respuesta será 437.

Nota

Un servidor no debería ofrecer a otro servidor artículos que ya le haya enviado antes o que ya hayan pasado por este último (el campo `Path` de la cabecera lo indica).

Esta situación se conoce como *doble ofrecimiento*.

13) Conexión desde un servidor esclavo (**SLAVE**)

Este comando sirve para informar al servidor de que la conexión proviene de un servidor esclavo. El servidor puede decidir, por ejemplo, dar más prioridad a esta conexión que a las que provienen directamente de clientes porque se supone que atiende a más usuarios. El código de respuesta es el 202.

```
SLAVE
```

14) Mensaje de ayuda (**HELP**)

Este comando se utiliza para obtener un mensaje de ayuda. El servidor responde con un código 100 y, a continuación, un texto informativo de los comandos que acepta, acabado con una línea con un carácter "`.`".

```
HELP
```

15) Cerrar la conexión (**QUIT**)

Este comando se utiliza para cerrar la conexión. El servidor responde enviando un código 205 y cerrando la conexión.

```
QUIT
```

La especificación RFC 977 permite que las implementaciones añadan otros comandos a los estándares; sin embargo, recomienda que el nombre de estos nuevos comandos empiece con `X` para evitar posibles conflictos con posteriores extensiones oficiales.

Algunos ejemplos de comandos implementados por muchos servidores son los siguientes:

- `XGTITLE`: confiere un título descriptivo de uno o más grupos.
- `XHDR` y `XOVER`: dan, respectivamente, el valor de un campo y un resumen de la cabecera de uno o más artículos.

ANOTACIONES

- `XPAT`: proporciona los valores de un campo de la cabecera que concuerdan con un patrón, junto con los números de los artículos correspondientes.

La tabla siguiente resume los códigos de respuesta del protocolo NNTP:

Tabla 15.

Códigos de respuesta	
Código	**Significado**
100	Mensaje de ayuda.
200	Servidor preparado; se permite enviar artículos.
201	Servidor preparado; no se permite enviar artículos.
202	Conexión desde servidor esclavo.
205	El servidor cierra la conexión.
211	Grupo seleccionado.
215	Lista de grupos.
220	Contenido del artículo.
221	Cabecera del artículo.
222	Cuerpo del artículo.
223	Identificador del artículo.
230	Lista de artículos nuevos.
231	Lista de grupos nuevos.
235	Artículo recibido.
240	Artículo enviado.
335	Preparado para recibir un artículo de otro servidor.
340	Preparado para recibir un artículo del cliente.
400	Servicio no disponible.
411	No existe el grupo.
412	No hay ningún grupo seleccionado.
420	No hay ningún artículo seleccionado.
421	No hay artículo siguiente.
422	No hay artículo anterior.
423	No hay ningún artículo con este número.
430	No hay ningún artículo con este identificador.
435	No se desea recibir el artículo.
436	Error en la recepción del artículo.

ANOTACIONES

Códigos de respuesta	
Código	**Significado**
437	Artículo rechazado.
440	No se permite enviar artículo.
441	No se ha podido enviar el artículo.
500	Comando desconocido.
501	Error de sintaxis en el comando.
502	Permiso denegado.
503	Error interno.

Actividad

Estableced una conexión con un servidor NNTP (por ejemplo, con `telnet` *`servidor`* `119`). Efectuad algunas operaciones como listar los grupos que haya (¡atención: en algunos servidores la lista puede tener más de 10.000 líneas!), entrar en un grupo, ver la cabecera de algún artículo, leer alguno entero, etc., y observad los códigos numéricos de respuesta que envía el servidor.

ANOTACIONES

21. Servicio hipermedia: WWW

21.1. Documentos hipermedia

El **servicio WWW** (*world wide web*) ofrece acceso a información multimedia, que puede incluir contenidos de diferentes tipos (texto, imágenes, audio, vídeo, etc.) y referencias a otros elementos de información, según el modelo de los sistemas hipertexto.

Un **sistema hipertexto** permite recorrer un documento de manera no necesariamente lineal o secuencial, sino siguiendo las referencias o enlaces que el usuario selecciona y saltando a la parte referenciada. Es decir, en lugar de seguir un único camino lineal, se puede realizar un recorrido por el documento pasando por los arcos de un grafo. Esta manera de acceder a la información se suele llamar familiarmente *navegar*.



Cuando se generaliza la funcionalidad hipertextual para que permita navegar por elementos de información multimedia y no sólo por texto, se suele hablar de **sistemas hipermedia**.

En la terminología WWW, cada uno de los elementos de información a los que se puede acceder se llama *recurso*. Los **documentos hipermedia** constituyen un caso particular de recurso que contiene información estructurada, posiblemente con diferentes tipos de contenido y enlaces a otros recursos.

En la estructura de un documento, en general, se pueden distinguir los elementos siguientes:

1) La **estructura lógica**, es decir, la división del contenido en partes que se encuentran en diferentes niveles, como capítulos, secciones, apartados, subapartados, títulos, párrafos, figuras, tablas, encabezamientos, notas, etc.

ANOTACIONES

2) La **estructura física**, es decir, la disposición de las partes del documento en el medio de presentación (por norma general, papel o pantalla): la división del documento en páginas y grupos de páginas, la distribución del área de cada página en zonas para cabeceras y pies, los márgenes, los espacios reservados a figuras, tablas, etc.

En el modelo general de procesado de los documentos estructurados, el autor crea la estructura lógica, con el contenido correspondiente, y puede incluir ciertas directrices para controlar el aspecto visual que deberá tener el documento. En numerosas ocasiones, a un determinado conjunto de directrices de este tipo se le da el nombre de *estilo*. A partir de esta información, un proceso denominado **de formateo** se encarga de generar la estructura física.

Existen diferentes **representaciones normalizadas de los documentos estructurados**, las cuales van destinadas a objetivos diferentes:

a) Representaciones que se centran en la especificación de la estructura lógica, como el **SGML** (*Standard Generalized Markup Language*, publicado en el estándar internacional ISO 8879).

Nota

SPDL está basado en otro lenguaje muy popular que sirve para representar documentos formateados: el **lenguaje PostScript**.

b) Representaciones que se centran en la especificación de la estructura física, como el **SPDL** (*Standard Page Description Language*, publicado en el estándar ISO/IEC 10180).

c) Representaciones que comprenden tanto la estructura lógica como la física, por ejemplo el **estándar ODA** (*Open Document Architecture*, publicado en el estándar ISO/IEC 8613 y en la serie de recomendaciones ITU-T T.410).

21.2. Marcado: el SGML

Nota

DSSSL es la sigla de *document style semantics and specification language*, publicado en el estándar ISO/IEC 10179

El SGML contiene la especificación de un lenguaje que sólo permite representar la estructura de un documento desde el punto de vista lógico. En principio, un usuario puede aplicar los estilos que quiera para visualizar el documento. Sin embargo, lo más habitual es que las aplicaciones que utilizan este lenguaje proporcionen una serie de reglas para interpretar la estructura del documento y, en particular, para formatearlo. Por otro lado, se ha publicado otro estándar, lla-

mado **DSSSL**, que permite definir estilos de presentación con las transformaciones necesarias para convertir un documento SGML en otra notación, como el SPDL.

Un ejemplo de aplicación basada en SGML es el **HTML** (*Hypertext Markup Language*) diseñado para especificar documentos hipermedia.

Nota

Actualmente, la especificación oficial del HTML está bajo el control de un grupo de empresas y organizaciones conocido como World Wide Web Consortium (W3C). Con anterioridad, la responsabilidad era de un grupo de trabajo de la Internet Engineering Task Force (IETF), que cesó sus actividades en 1996 tras publicar el documento RFC 1866 con la especificación HTML 2.0.

El SGML se publicó en 1986 con el objetivo de facilitar la especificación de documentos estructurados. El lenguaje definido en este estándar, sin embargo, es lo suficientemente general para permitir la representación de muchos otros tipos de información estructurada (bases de datos, etc.).

Lectura complementaria

Podéis ampliar la información sobre el SGML en la obra siguiente:

Charles Goldfarb (1990). *The SGML Handbook*. Oxford: Oxford University Press.

21.3. Transferencia de hipermedia: el HTTP

El HTTP (*hipertext transfer protocol*) es la base del servicio WWW. Sus orígenes fueron los trabajos de un grupo de investigadores del CERN (Centro Europeo de Investigación Nuclear) que, al final de los años ochenta y principios de los noventa, definieron un protocolo para acceder de manera sencilla y cómoda a la información distribuida en las diferentes sedes del centro. Cuando el conjunto de sistemas accesibles con este mecanismo se extendió a otras organizaciones y países de todo el mundo, nació lo que hoy conocemos como WWW (*World Wide Web*).

De la misma manera que el HTML, el HTTP también evoluciona constantemente a medida que los implementadores incorporan nuevas funcionalidades para adaptarlo a diferentes tipos de aplicaciones particulares. En 1996, se publicó el documento RFC 1945, en que se define la versión

ANOTACIONES

HTTP/1.0 (este documento también contiene las especificaciones de la denominada *versión 0.9* para facilitar la interoperabilidad con implementaciones antiguas). Sin embargo, el año siguiente ya se publicó la especificación HTTP/1.1 en el documento *RFC 2068*.

A continuación, veremos las características principales de la funcionalidad que ofrece el HTTP. Sin embargo, antes de entrar en los detalles del protocolo, estudiaremos el método general utilizado en el servicio WWW para identificar la información a la que se quiere acceder.

21.3.1. Direccionamiento: identificadores uniformes de recurso (URI)

Nota

La definición de URI constituye otro ejemplo de especificación en constante evolución, al menos hasta que se publicó el documento *RFC 2396*, en 1998.

Desde un documento se pueden referenciar recursos especificando sus direcciones, que se representan por medio de lo que en la terminología WWW se denomina *identificador uniforme de recurso* o *URI*.

La forma general de un URI se puede expresar del modo siguiente:

```
esquema: identificador
```

Ésta es la forma correspondiente a un **URI absoluto**. La palabra que se encuentra antes del separador "`:`" es el esquema, que determina la sintaxis del identificador. Esta sintaxis también puede permitir la especificación de **URI relativos**, en que se omite el esquema y el separador "`:`".

Un URI puede ser un **localizador** (**URL**), si especifica cómo se accede al recurso, y/o un **nombre** (**URN**) si identifica el recurso por medio de un conjunto de atributos:

```
ftp://[usuario[:contraseña]@]servidor/camino
```

Según el esquema, éstas son algunas sintaxis posibles de los URL:

a) **`ftp`**

Este tipo de URL identifica un recurso accesible por medio del protocolo FTP: `servidor` es el nombre del ordenador servidor

ANOTACIONES

y `camino` es el nombre completo de un fichero o directorio. Si el nombre de usuario no se encuentra presente, significa que es preciso utilizar `anonymous` y, como contraseña, cualquier cadena de caracteres.

b) **`news`**

Este URL identifica un conjunto de artículos si `identificador` es el nombre de un grupo de noticias Usenet, en el caso de que sólo sea un identificador de mensaje (campo `Message-ID`), entonces sólo identificará un determinado artículo:

```
news: identificador
```

c) **`mailto`**

Este identificador URL representa una dirección de correo electrónico:

```
mailto: dirección
```

d) **`telnet`**

Este URL representa un servicio accesible por medio del protocolo TeInet en el servidor y el puerto especificados:

```
telnet://[usuario[:contraseña]@]servidor[:puerto][/]
```

e) **`http`**

Cuando deba accederse al recurso identificado por medio del HTTP, se utilizará un URL de acuerdo con la sintaxis siguiente:

```
URL-HTTP = URL-HTTP-absoluto | URL-HTTP-relativo
URL-HTTP-absoluto = http://servidor[:puerto]
[camino-absoluto]
URL-HTTP-relativo = camino-absoluto | camino-relativo
camino-absoluto = /camino-relativo
camino-relativo = [camino] *(;parámetro)[? consulta]
[# fragmento]
camino = segmento *(/segmento)
```

ANOTACIONES

En esta sintaxis, `servidor` es el nombre del ordenador con que es preciso establecer la conexión HTTP para acceder al recurso, y `puerto` es el número de puerto en que debe efectuarse la conexión que, por defecto, es el número asignado al servicio WWW, es decir, el 80.

Nota

`segmento`, `parámetro`, `consulta` o `fragmento` pueden ser una cadena vacía.

Cada uno de los segmentos que forman el `camino` se asocia, por norma general, a un nombre de directorio o de fichero. Estos nombres no pueden incluir los caracteres "`/`", "`;`", "`?`", "`\`", "`#`", "`"`", "`<`", "`>`" o espacio. El carácter "`/`" sí que puede estar presente en un parámetro, y tanto "`/`" como "`;`" y "`?`" también pueden estar presentes en los componentes `consulta` y `fragmento`.

Si en el `camino` aparece el carácter "`%`", debe ir seguido de dos dígitos hexadecimales, y esta secuencia de tres caracteres equivale al carácter que tenga el código indicado por los dígitos. Por tanto, se puede utilizar una secuencia "`%HH`" para representar cualquiera de los caracteres no permitidos en un URL, incluyendo el mismo carácter "`%`" (que se representaría con "`%25`").

Nota

A continuación, presentamos diferentes ejemplos de URL con esquemas diferentes (lo dos últimos son URL HTTP relativos):

```
ftp://ftp.uoc.es/pub/doc/README
news:comp.infosystems.www.misc
mailto:Ernest.Udiant@campus.uoc.edu
http://www.uoc.es/
http://www.acme.com/%7Eadmin/
http://www.acme.com/buscador/busca.cgi?nom=Internet
http://www.acme.com/doc/ayuda.html#buscador
/doc/ayuda.html#buscador
ayuda.html#buscador
```

Los URL relativos se pueden utilizar en documentos HTML, en concreto en los atributos que representan direcciones de otros recursos (`HREF`, `SRC`, `ACTION`). Estos URL se consideran relativos a una determinada dirección base, que es la indicada por el elemento `BASE` del documento o, si este elemento no se encuentra presente, la dirección que se ha utilizado para acceder al mismo documento HTML.

ANOTACIONES

Para obtener el correspondiente URL absoluto, es preciso aplicar las reglas siguientes:

1) Si el URL relativo contiene un camino absoluto, sustituye al camino de la dirección base.

2) Si el URL relativo contiene un camino relativo (y el camino no está vacío), se sustituye el último segmento de la dirección base (la parte del camino que se encuentre después del último "/") por éste. Como caso especial, cada segmento con nombre ".." que se encuentre en el inicio del camino relativo debe suprimirse junto con el último segmento que quede al final del camino de la dirección base.

3) Si el camino del URL relativo está vacío, pero hay alguno de los otros componentes (parámetros, consulta, fragmento), se toma el camino de la dirección base y se sustituyen los componentes correspondientes por los del mismo.

Nota

Si el URL relativo está completamente vacío, se toma directamente toda la dirección base, incluyendo el fragmento, si hay (en los otros casos, el fragmento de la dirección base no se considera).

Nota

Interpretación del URL relativo

Dada la dirección base `http://www.uoc.edu/extern/ct/home/home.html`, los URL relativos siguientes deberían interpretarse de este modo:

Tabla 16.

URL relativo	URL absoluto
/accesset98/	http://www.uoc.edu/accesset98/
otras/pagina.html	http://www.uoc.edu/extern/ct/home/otras/pagina.html
../../logo.jpeg	http://www.uoc.edu/extern/logo.jpeg
#final	http://www.uoc.edu/extern/ct/home/home.html#final

21.3.2. Conceptos básicos del HTTP

El HTTP sigue el modelo general de peticiones y respuestas entre un cliente y un servidor. Se basa en un servicio de transporte fiable; pero, es independiente del mecanismo de transporte concreto utilizado. No obstante, lo más habitual es que cliente y servidor se comuniquen por

ANOTACIONES

medio del TCP. En este caso, el puerto por defecto para establecer las conexiones es el asignado oficialmente al servicio WWW, es decir, el 80.

En el HTTP/1.0, el cliente establece una conexión con el servidor y le envía un mensaje HTTP con la petición; y, a continuación, el servidor envía al cliente otro mensaje HTTP con la respuesta y cierra la conexión. Si quiere efectuar más peticiones, el cliente debe establecer una nueva conexión para cada una. En el HTTP/1.1, en cambio, es posible intercambiar diferentes peticiones y respuestas en una misma conexión que se denomina *conexión persistente*. Éste es el modo de funcionamiento por defecto en el HTTP/1.1.

Un **mensaje HTTP** consta de una primera línea, en la que hay información específica del protocolo, seguida de un mensaje con el mismo formato que los mensajes de correo electrónico, según la especificación RFC 822. Es decir, después de la primera línea debe haber una cabecera formada por una serie de campos, una línea en blanco y un cuerpo. En casos particulares, la cabecera y/o el cuerpo pueden estar vacíos, pero la línea en blanco que los separa siempre debe estar presente.

El cuerpo del mensaje, junto con los campos de la cabecera que proporcionan información sobre su contenido, forman lo que en el HTTP se denomina una *entidad*. Cada entidad corresponde a un recurso.

Dependiendo de si el mensaje HTTP es una petición o una respuesta, la primera línea recibe el nombre de *línea de petición* o *línea de estatus*, respectivamente.

La **sintaxis de una línea de petición** es la siguiente:

```
método <SP> URI <SP> versión <CRLF>
```

En esta línea, `método` especifica qué tipo de operación (o, en la terminología HTTP, qué método) solicita el cliente, `URI` identifica el recurso a que se debe aplicar la operación y `versión` debe ser la cadena `HTTP/1.0` o `HTTP/1.1`, según la versión del protocolo. El URI debe ser un URL HTTP relativo que contenga un camino absoluto; es decir, que empiece por "/" (excepto cuando el servidor sea un *proxy*, como veremos más adelante).

ANOTACIONES

Nota

Siempre que un usuario quiera utilizar un cliente para obtener el recurso dado por la dirección `http://www.acme.com:8000/doc/principal.html`, el cliente debe conectarse al puerto 8000 del servidor `www.acme.com` y enviarle una petición que empiece con esta línea:

```
GET /doc/principal.html HTTP/1.0
```

En este caso, el método utilizado es `GET` (obtener recurso). Ahora bien, si la dirección fuera `http://www.acme.com:8000/`, la línea que debería enviarse sería simplemente la siguiente:

```
GET / HTTP/1.0
```

La **sintaxis de las líneas de estatus** es la siguiente:

```
versión <SP> código <SP> frase <CRLF>
```

La cadena `versión` es igual que en la línea de petición, el código es un número de tres dígitos, el primero de los cuales indica de qué tipo de respuesta se trata, y `frase` es un texto explicativo en formato libre, inteligible para los usuarios humanos.

Los posibles **códigos de una respuesta HTTP/1.0** son los siguientes:

Tabla 17.

Código	Significado
200	Operación efectuada.
201	Se ha creado un nuevo recurso.
202	Petición aceptada.
204	La respuesta está vacía.
301	El recurso solicitado ha cambiado de dirección.
302	El recurso solicitado ha cambiado temporalmente de dirección.
304	El contenido del recurso no ha cambiado.
400	Petición incorrecta.
401	Usuario no autorizado.

Nota

En el HTTP, el significado del primer dígito del código de respuesta varía un poco de lo que se considera común (que encontraréis en el anexo 4) y es el siguiente:

- 1: Información.
- 2: Éxito.
- 3: Redireccionamiento.
- 4: Error en el cliente.
- 5: Error en el servidor.

Código	Significado
403	Acceso prohibido.
404	No se ha encontrado el recurso.
500	Error interno del servidor.
501	Operación no implementada.
502	Error en otro servidor.
503	Servicio no disponible en la actualidad.

Nota

En el HTTP/1.1 se han añadido a esta lista hasta 22 nuevos códigos de respuesta. No obstante, si un cliente recibe un código *xyz* que no conoce, debe interpretarlo como si fuera x00. En este sentido, el código 100 significa 'continuar' y el código 300 quiere decir 'recurso accesible en una dirección o más'.

Almacenamiento local de respuestas (memoria caché)

Para optimizar el tráfico de la red, es habitual que los clientes HTTP guarden una copia de las respuestas que reciben de los servidores, junto con el URL que han utilizado en la petición. El almacén destinado a este fin recibe el nombre de **memoria caché**. Si el usuario quiere volver a acceder a un recurso que ya había solicitado previamente, el cliente puede utilizar la copia de la memoria caché en lugar de volver a enviar la misma petición al servidor.

El cliente puede decidir, o el usuario puede configurar, el tiempo que se guardarán los elementos de la memoria caché, su dimensión máxima o la política que deberá seguirse para borrar sus elementos cuando esté llena. Asimismo, el HTTP también proporciona directrices para determinar qué recursos se pueden guardar en la memoria caché y cuáles no, o cuánto tiempo se puede guardar como máximo un determinado recurso.

Sin embargo, no sólo los clientes pueden guardar copias de las respuestas obtenidas de los servidores; también pueden hacerlo los servidores intermediarios o *proxies*.

ANOTACIONES

Nota

Consultad los servidores intermediarios en el apartado 21.3.4 de esta unidad.

Cabeceras de los mensajes HTTP

Los campos que puede haber en la cabecera de un mensaje HTTP se pueden clasificar en cuatro grupos:

1) Campos generales que pueden estar presentes tanto en los mensajes de petición, como en los de respuesta.

2) Campos referentes a la entidad contenida en el cuerpo del mensaje y que también pueden estar presentes en peticiones y respuestas.

3) Campos propios de las peticiones.

4) Campos propios de las respuestas.

El HTTP proporciona un mecanismo de extensión que permite incluir campos no estándar en las cabeceras. Si una implementación no reconoce un determinado campo, lo que debe hacer es simplemente ignorarlo (o retransmitirlo tal como lo encuentra si es un servidor intermediario).

Los campos que puede contener la cabecera son los siguientes:

1) **Campos generales definidos en HTTP/1.0:**

a) **`Date:`** ***`fecha`***. Indica la fecha y la hora en que se originó el mensaje. Los valores de los campos HTTP que representan fechas se deben expresar de una de estas tres maneras, siempre referidas al tiempo universal (`TU`, anteriormente conocido como **hora del meridiano de Greenwich** o `GMT`):

```
Fri, 06 Dec 2002 09:22:15 GMT (RFC 1123, forma recomendada)
Friday, 06-Dec-2002 09:22:15 GMT (RFC 850)
Fri Dec 6 09:22:15 2002 (formato ANSI C)
```

b) **`Pragma:`** ***`1#directriz`***. Este campo sirve para proporcionar una lista de directrices a los sistemas que intervienen en la transferencia (clientes o servidores). La semántica de las directrices depende de las implementaciones; sin embargo, hay una definida por el protocolo, que se representa con la palabra `no-cache`. Si un mensaje de petición contiene esta directriz, significa

ANOTACIONES

que debe enviarse la petición al servidor correspondiente aunque exista una copia de la respuesta guardada en la memoria caché.

Nota

La especificación HTTP/1.1 añade cinco nuevos campos generales:

- `Cache-Control`: directrices sobre la política de memoria caché.
- `Connection`: opciones de conexión. La opción `close` indica que el emisor cerrará la conexión después de que se haya enviado la respuesta.
- `Transfer-Encoding`: codificación aplicada al cuerpo del mensaje.
- `Upgrade`: versiones del protocolo soportadas.
- `Via`: intermediarios por los que ha pasado el mensaje.

2) Campos referentes a la entidad definidos en el HTTP/1.0:

a) **`Content-Length:`** ***`1*digito`*****.** Indica la longitud en bytes del cuerpo de la entidad. Si este campo no está presente en una respuesta, el cliente sólo puede saber cuál es el final del cuerpo cuando el servidor cierra la conexión. En un mensaje de petición que incluya un cuerpo, el campo `Content-Length` es obligatorio, puesto que de otro modo el servidor no podría enviar la respuesta y no se podría cerrar la conexión.

b) **`Content-Type:`** ***`tipo`*****.** Indica el tipo de contenido del cuerpo de la entidad (por ejemplo, `text/html` si es un documento HTML). El valor de este campo se representa según la especificación `MIME` y, por tanto, puede incluir parámetros (por ejemplo, `charset=ISO-8859-1` si los caracteres están codificados según el alfabeto latín1). Todos los mensajes HTTP con cuerpo deberían incluir el campo `Content-Type`.

ANOTACIONES

Nota

Consultad los nuevos campos de cabeceras en el subapartado 19.5.1.

c) **`Content-Encoding:`** ***`codificación`***. Indica si se ha aplicado alguna transformación al contenido de la entidad (por ejemplo: `x-gzip`). El valor de este campo se representa según la especificación `MIME`.

d) **`Last-Modified:`** ***`fecha`***. Indica la fecha y la hora en que el recurso contenido en el cuerpo se modificó por última vez.

e) **`Expires:`** ***`fecha`***. Indica la fecha y la hora a partir de la cual el contenido del cuerpo se puede considerar obsoleto o caducado al efecto de su almacenamiento en la memoria caché. La presencia de este campo significa que, posiblemente, el recurso se modificará en la fecha indicada o dejará de existir; sin embargo, no implica que los cambios, si se producen, deban efectuarse necesariamente en esta fecha. Ahora bien, si la fecha de caducidad es igual o anterior a la especificada en el campo `Date`, la entidad no debe almacenarse en la memoria caché.

f) **`Allow:`** ***`1#método`***. Indica los métodos HTTP que se pueden aplicar al recurso solicitado.

Nota

La especificación HTTP/1.1 añade seis nuevos campos de entidad adicionales:

- `Content-Base`: dirección base para interpretar URL relativos.
- `Content-Language`: lenguaje que se ha utilizado.
- `Content-Location`: URI de la entidad, en caso de que el recurso correspondiente disponga de más de una.
- `Content-MD5`: secuencia de bits para comprobar la integridad del contenido.
- `Content-Range`: por si una entidad se envía en diferentes fragmentos.
- `ETag`: etiqueta asociada a la entidad, por si el recurso dispone de más de una.

ANOTACIONES

3) Campos propios de las peticiones HTTP/1.0:

a) **`From:`** ***`dirección`***. Este campo contiene la dirección de correo electrónico del usuario que solicita el recurso.

b) **`User-Agent:`** ***`1*implementación`***. Este campo permite identificar la implementación del cliente. Se expresa como una lista de "productos" (por ejemplo, tipo de navegador, librerías de soporte que utiliza, etc.) con números de versión opcionales (separados de los nombres con "/"), que puede incluir comentarios entre paréntesis. El servidor puede utilizar esta información para generar estadísticas, detectar qué implementaciones producen ciertos errores, adaptar las respuestas al tipo de cliente, etc.

Ejemplo

Ejemplos de listas de productos:

- `Lynx/2.8rel.2 libwww-FM/2.14.`
- `NCSA_Mosaic/2.6 (X11;SunOS 4.1.4 sun4m) libwww/2.12.`
- `Harvest/1.5.17.`

c) **`Referer:`** ***`URI`***. Si el usuario selecciona un enlace de un documento HTML o de otro recurso que tiene dirección propia, el cliente puede incluirla en el campo `Referer` del mensaje de petición. El servidor puede utilizar esta información para generar estadísticas, detectar desde qué recurso se referencia una dirección incorrecta u obsoleta, etc.

d) **`If-Modified-Since:`** ***`fecha`***. Cuando el cliente ya dispone de una copia del recurso solicitado en la memoria caché, puede utilizar este campo para llevar a cabo una operación `GET` condicional: si el recurso se ha modificado con posterioridad a la fecha indicada, se efectuará la operación de manera normal y, si no, el servidor enviará una respuesta sin cuerpo y con el código 304. Este campo sólo es aplicable al método `GET`.

e) **`Authorization:`** ***`esquema#parámetro`***. Con este campo, un usuario puede presentar sus credenciales a un servidor (por norma general, un nombre y una contraseña) para que le permita acceder a recursos no públicos, es decir, de acceso restringido. La primera parte de la credencial indica el esquema de autenticación a utilizar. El HTTP/1.0 sólo define un esquema denominado *básico*, pero se puede utilizar cualquiera mientras sea conocido por el cliente y el servidor. Si se utiliza el esquema básico, el valor de este campo debe ser la palabra `Basic` seguida de la cadena de caracteres que resulta de codificar en `Base64` el nombre de usuario y su contraseña separados por "`:`".

Nota

Consultad la codificación en `Base64` en el apartado 19.5.1.

ANOTACIONES

Ejemplo

Si el nombre de usuario es `webmaster` y su contraseña es `Secret`, se codificará en `Base64` la cadena `webmaster:Secret`. El campo quedaría de este modo:

```
Authorization:Basic d2VibWFzdGVyOlNlY3JldA==
```

Obviamente, la descodificación de este campo es trivial y, por tanto, no hay protección contra terceros que tengan la posibilidad de inspeccionar los mensajes y deseen obtener un acceso no autorizado. Para conseguir que el método de autenticación sea más seguro, se puede utilizar un esquema más sofisticado o alguna variante del protocolo basada en el cifrado de los datos, por ejemplo el HTTPS.

La respuesta a una petición que incluya el campo `Authorization` no debería guardarse en la memoria caché.

Nota

En la especificación HTTP/1.1 se añaden los campos de petición siguientes:

- `Accept`: tipo de contenido que el cliente puede aceptar.
- `Accept-Charset`.
- `Accept-Encoding`.
- `Accept-Language`.
- `Host`: nombre del servidor a quien va dirigida la petición, por si tiene más de uno; este campo, obligatorio en las peticiones HTTP/1.1, permite que un ordenador con diferentes nombres actúe como si fuera diferentes servidores al mismo tiempo.

ANOTACIONES

- `If-Match`: permite comparar entidades por sus etiquetas.
- `If-None-Match`.
- `If-Range`.
- `If-Unmodified-Since`.
- `Max-Forwards`.
- `Proxy-Authorization`.
- `Range`: sirve para solicitar un fragmento de una entidad.

4) Campos propios de las respuestas HTTP/1.0:

a) **`Server:`** ***`1*implementación`***. Este campo es similar a `User-Agent`, pero referido al servidor.

b) **`Location:`** ***`URI`***. Indica que el recurso solicitado debe obtenerse en otra dirección. Éste será el caso cuando el código de respuesta sea 301 ó 302. Por norma general, cuando el cliente reciba una respuesta que incluya este campo, generará de manera automática una nueva petición con la dirección indicada.

c) **`WWW-Authenticate:`** ***`1#(esquema`*** **`realm = "`*****`reino`*****`" *( ,`** ***`parámetro`*** **`))`**. Este campo es obligatorio cuando el código de las respuestas es 401. Indica que es necesario presentar una credencial para acceder al recurso solicitado, o que la que se ha presentado no es válida.

Un servidor puede agrupar sus recursos de acceso restringido en reinos, cada uno con su esquema de autenticación y conjunto de usuarios autorizados (con una misma credencial debería poderse acceder a todos los recursos de un reino). El valor de este campo es una lista con los nombres de los reinos aplicables al recurso solicitado, junto con sus esquemas y parámetros opcionales para poder llevar a cabo la autenticación con el campo `Authorization` de la petición.

ANOTACIONES

Nota

En la especificación HTTP/1.1 se añaden los campos de respuesta siguientes:

- `Age`.
- `Proxy-Authenticate`.
- `Public`: lista de métodos soportados por el servidor.
- `Retry-After`.
- `Vary`: lista de campos de la petición que se han utilizado para seleccionar una entidad, cuando el recurso dispone de más de una.
- `Warning`: información adicional sobre el estatus de la respuesta.

21.3.3. Métodos del servicio HTTP

El protocolo HTTP/1.0 define los tres métodos siguientes:

1) **Método `GET`.** Este método sirve para obtener la entidad correspondiente al URI especificado en la línea de petición. Por norma general, el servidor traducirá el camino del URI a un nombre de fichero o de programa:

- En el primer caso, el cuerpo de la entidad será el contenido del fichero.
- En el segundo caso, el servidor ejecutará el programa y la entidad será el resultado que genere.

Los componentes `parámetro` y/o `consulta` del URI se pueden utilizar como argumentos del programa.

Nota

Cuando el camino de un URI identifica un programa, la manera como se le pasan los valores de los parámetros o las consultas es un asunto local del servidor. Por ejemplo, un mecanismo utilizado habitualmente es el denominado CGI (*Common Gateway Interface*).

2) **Método `HEAD`.** Este método es igual que el `GET`, excepto que en la respuesta el cuerpo estará vacío y, por tanto, sólo tendrá cabecera (que deberá ser idéntica a la que se habría enviado si el método fuera el `GET`). Por norma general, se utiliza el método `HEAD`,

ANOTACIONES

por ejemplo, para comprobar si un URL es válido o para obtener información sobre un recurso sin necesidad de transferir su contenido.

3) **Método POST.** Este método sirve para enviar una entidad que el servidor debe incorporar al recurso identificado por el URI de la línea de petición. La semántica de este método depende del tipo de recurso de que se trate. Por ejemplo, se puede utilizar para añadir contenido a un recurso existente, para enviar un mensaje a un tablón de anuncios o un grupo de noticias, para crear un nuevo registro en una base de datos, para pasar datos a un programa que debe ejecutarse en el servidor, etc. Un caso típico de este último ejemplo son los datos de un formulario HTML.

El código de respuesta a una operación `POST` por norma general será 201, si como resultado, se ha creado un nuevo recurso (en este caso, el cuerpo de la respuesta debería contener una referencia a este recurso), o bien 200 ó 204 si no se ha creado ninguno (el cuerpo de una respuesta 200 contendrá una descripción del resultado obtenido, y el de una respuesta 204 simplemente estará vacío).

Una propiedad del método `POST` es que, si se envían dos peticiones iguales, el resultado de la segunda no debe ser necesariamente el mismo que el de la primera. Por tanto, la respuesta a una operación `POST` no debería guardarse en la memoria caché.

Nota

En el HTTP/1.1 se han añadido una serie de métodos nuevos, entre lo cuales, los siguientes:

- `PUT`: para crear un recurso con el URI especificado en la petición.
- `DELETE`: para borrar un recurso.
- `OPTIONS`: para obtener información sobre las opciones de transferencia.
- `TRACE`: para obtener una copia del mensaje como ha llegado a su destino final.

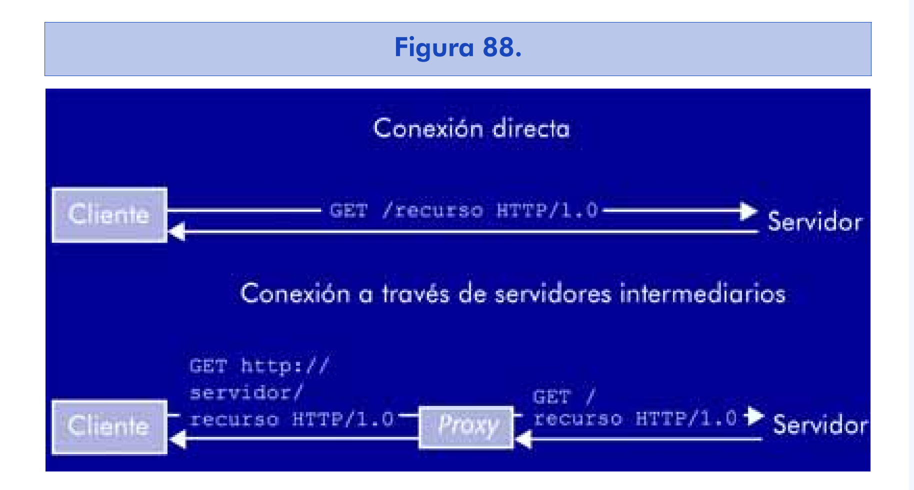
Actividad

Estableced una conexión con un servidor HTTP con `telnet servidor 80` y averiguad la fecha en que ha sido modificada por última vez la página principal, o alguna otra información referida al servidor (con un mensaje formado por la línea `HEAD / HTTP/1.0`, una cabecera vacía, una línea en blanco y un cuerpo vacío).

21.3.4. Intermediarios: *proxies* y pasarelas

El modelo general del HTTP no sólo considera la conexión directa de un cliente con el servidor, sino también la conexión por medio de servidores intermediarios, como es el caso de los *proxies*:

Figura 88.



En la configuración con *proxy*, el cliente establece la conexión con el servidor *proxy* que, a su vez, establece otra conexión, como cliente, con el servidor final. Cuando el *proxy* recibe el mensaje de petición del cliente, puede generar una respuesta propia o retransmitirla al servidor final. En el segundo caso, el *proxy* puede introducir modificaciones en la petición, según la aplicación para la que esté diseñado, y, cuando reciba la respuesta del servidor final, la retransmitirá al cliente, también con la posibilidad de efectuar cambios.

Asimismo, es posible que haya más de un *proxy* en la cadena; es decir, que el primer *proxy* no se conecte directamente al servidor final, sino por medio de otro *proxy*, y así sucesivamente.

En las peticiones que un cliente (u otro *proxy*) envía a un *proxy*, existe una variación respecto al caso de la conexión directa de cliente a servidor: el URI de la línea de petición no debe ser un URL HTTP relativo, sino que debe ser un URI absoluto. De lo contrario, el *proxy* no sabría cuál es el servidor final al que va destinada la petición.

El uso de un servidor *proxy* tiene diferentes aplicaciones. Las principales son las siguientes:

a) Actuar como cortafuegos que aísle una red local del resto de las redes. En esta configuración, los clientes no tienen acceso directo al exterior de su red, y toda comunicación con los servidores remotos tiene lugar por medio del *proxy*. Ello permite minimizar el riesgo de que usuarios externos comprometan la seguridad de los sistemas locales (accesos no autorizados, sabotajes, etc.).

b) Tener una memoria caché compartida entre los usuarios de la red local. Si diferentes clientes solicitan directamente un mismo recurso, por norma general guardarán la misma copia de la respuesta en sus respectivas memorias caché. Si lo solicitan por medio de un *proxy*, la primera petición necesitará un acceso al servidor remoto; sin embargo, las siguientes quizá puedan aprovechar la copia ya guardada en la memoria caché, aunque provengan de clientes diferentes.

c) Construir una jerarquía de memorias caché de *proxies*: en el nivel más bajo se encuentran los *proxies* a que acceden directamente los clientes, en un segundo nivel existen los *proxies* a que acceden los del primer nivel, etc. Incluso puede haber *proxies* a escala de todo un país. Hay un protocolo denominado **ICP** (*Internet cache protocol*) que permite coordinar los diferentes servidores *proxy* de una jerarquía.

Nota

Además de los *proxies*, existe otro tipo de servidor intermediario llamado **pasarela**. Las pasarelas actúan como si fueran el servidor final, con la diferencia respecto al *proxy* de que el cliente no sabe que no se conecta directamente al servidor final. Una pasarela se puede utilizar como cortafuegos en la red del servidor o para traducir las peticiones HTTP de manera que se pueda acceder a recursos almacenados en sistemas no HTTP.

ANOTACIONES

22. Mensajería instantánea

No cabe duda de que Internet ha cambiado la manera de comunicarnos. Para muchos de nosotros, el correo electrónico ha reemplazado casi por completo al correo tradicional y ha hecho disminuir de forma considerable las llamadas telefónicas que realizamos. Diariamente se envían millones de correos electrónicos.

A pesar de lo rápido que es el sistema de correo, aún no lo es bastante. En algunos servidores, se pueden acumular retardos de algunos minutos. Además, por rápido que sea, se trata de un sistema asíncrono, que no permite mantener una conversación fluida. Cada mensaje enviado o recibido implica una serie de pasos a seguir, botones a pulsar y ventanas a abrir. Por todo ello, se han popularizado los sistemas de mensajería instantánea, un clásico del modelo *peer-to-peer*.

Las aplicaciones de mensajería instantánea permiten mantener una lista de personas (las *buddy lists* o listas de contactos) con las que uno desea mantenerse en contacto y mandarles mensajes, siempre y cuando estén concetados. Este detalle no nos debe preocupar, porque es la propia aplicación la que se encarga de verificarlo.

La manera de mandar mensajes es muy simple. La aplicación nos presenta una ventana en la que tecleamos el contenido del mensaje y donde se verá lo que nos conteste el interlocutor. De esta manera, se puede mantener una conversación en tiempo real.

Nota

Algunos sistemas permiten el uso de cámaras de vídeo, micrófonos y altavoces para que la conversación no se circunscriba sólo a texto escrito.

Además de la conversación básica, la mayoría de programas de mensajería instantánea ofrecen otras prestaciones, como las siguientes:

- Chat que permite establecer conversaciones en grupo.
- Pizarra que permite ver dibujos realizados por el interlocutor, en el momento o grabados en archivos.
- Archivos que permite compartir archivos.

ANOTACIONES

Desde que en 1983 apareció el primer sistema de mensajería instantánea, han salido al mercado otros muchos productos, algunos usando protocolos propietarios, otros intentando buscar un estándar. De hecho, la IETF ha recopilado muchas de estas contribuciones y está desarrollando el *Instant Messaging and Presence Protocol*, por el cual se espera que pase el futuro de la mensajería instantánea.

Nota

En general, el honor de haber creado el primer sistema de mensajería instantánea se atribuye a Mirabilis, la compañía formada por cuatro estudiantes israelíes que en 1996 lanzó ICQ. Pero no debemos olvidar que en los sistemas Unix existe desde 1983 una aplicación llamada `talk` que permite la conversación entre dos usuarios conectados a sendos sistemas Unix via `telnet` o `rlogin`

Seguramente, la proliferación de los ordenadores domésticos, Internet y la web de finales de los noventa, mayoritariamente en entorno Windows, ha eclipsado un tanto todo lo que tenía que ver con el entorno Unix.

22.1. Programas de mensajería instantánea

Sin pretender crear una lista exhaustiva, vamos a repasar los principales programas que proporcionan la funcionalidad de la mensajería instantánea en un entorno *peer-to-peer*. Algunos de los programas presentados son de software libre, pero otros no.

22.1.1. ICQ

ICQ fue el primer programa que apareció (en 1996) y actualmente aún es de los más populares. Las siglas responden a la frase "*I seek you*", ('te estoy buscando'). ICQ usa control centralizado a través de servidores, y la comunicación entre clientes y servidores, y entre clientes se realiza mediante un protocolo propietario (ICQ, v.5).

Nota

La dirección de la web de ICQ es la siguiente: http://www.icq.com.

ANOTACIONES

22.1.2. AIM

AIM (AOL *Instant Messenger*), que apareció poco después de ICQ, es la evolución de las comunidades que America OnLine ya tenía desarrolladas con otras tecnologías más antiguas, como las BBS (*Bulletin Board System*). AIM también usa un protocolo propietario. Por tanto, el cliente tiene que ser específico para este entorno.

Nota

En 1998, AOL compró Mirabilis e integró ICQ en su suite de servicios ofrecidos.

Nota

La dirección de la web de AIM es la siguiente: http://www.aim.com.

22.1.3. MSN Messenger

Microsoft también ha entrado en el mundo de los sistemas de mensajería instantánea con MSN (Microsoft Network) Messenger. Es muy parecido a los dos descritos anteriormente.

Nota

La dirección de la web de MSN Messenger es la siguiente: http://messenger.msn.com

22.1.4. Jabber

Jabber es más que un mero sistema de mensajería. Es un conjunto de protocolos y tecnologías que permiten el desarrollo de múltiples aplicaciones *peer-to-peer*. Permite desarrollar clientes, servidores, componentes, bibliotecas, etc. Está basado en XML y es la base de los protocolos que la IETF está en proceso de estandarización.

Nota

La dirección de la web de Jabber es la siguiente: http://www.jabber.org.

Sus autores presentan entre las ventajas de Jabber las siguientes: permite entornos descentralizados, es abierto, público, seguro, extensible y flexible.

22.1.5. GAIM

Gaim es un cliente de mensajería instantánea multiprotocolo, para GNU/Linux, Unix BSD, MacOS X y Windows. Es compatible con AIM, ICQ, MSN Messenger, Yahoo, IRC, Jabber, Gadu-Gadu y Zephyr.

Nota

La dirección de la web de GAIM es la siguiente: http://gaim.sourceforge.net.

Una de sus mejores prestaciones es que permite estar conectado con diferentes redes de mensajería simultáneamente. Así, un usuario de

Gaim puede estar en un chat de AOL mientras está conversando con un amigo de Yahoo y dentro de un canal IRC, todo en la misma aplicación.

ANOTACIONES

Resumen

En este curso hemos presentado las bases de funcionamiento de la red Internet.

En la primera parte se han presentado los conceptos básicos de redes de computadores y los diferentes sistemas de comunicaciones que se utilizan hoy día, siguiendo un hilo conductor histórico, que permite entender el porqué de muchas de las limitaciones y particularidades que poseen. Se ha expuesto, también, la arquitectura de protocolos, un concepto básico en redes de computadores. Como paradigma de dicho concepto, se ha explicado el modelo de referencia OSI y se han descrito los siete niveles que lo forman. Las arquitecturas reales, en particular Internet, suelen describirse y explicarse en relación con el modelo OSI.

En la segunda parte hemos hecho un repaso, desde un punto de vista descriptivo, a las redes de área local. El principio básico de funcionamiento de las redes de área local es la difusión de tramas en un medio compartido. Así, las estaciones de la red, cuando quieran establecer una comunicación con otra (u otras) estaciones, deberán generar tramas de bytes a partir de los datos que quieran transferir. Y ponerlas en el medio para que lleguen a su o sus destinatarios.

Se han presentado las topologías más usuales, destacando las topologías en bus y en anillo. En un bus, todas las estaciones se conectan a un mismo cable, mientras que en un anillo, cada estación está conectada a dos más, creando una estructura circular. Ambas topologías tienen ventajas e inconvenientes, siendo el más importante de estos la dificultad de mantenimiento y localización de averías que presentan ambas. Esto ha propiciado el desarrollo del cableado estructurado, como mecanismo para instalar redes de área local más fiables y siguiendo estándares universales.

Asociados al cableado estructurado han aparecido dispositivos muy comunes hoy en día como son los concentradores y los conmutado-

ANOTACIONES

res. Los primeros simulan un bus y los segundos aprovechan el concepto de conmutación para conseguir redes más eficientes. Siempre que se puede, las tramas no se difunden a todas las estaciones presentes en la red, si no sólo a sus destinatarias.

Finalmente se han presentado las políticas de acceso al medio más habituales, las cuales van asociadas a la topología usada. Así, hemos visto la política de paso de testigo como la más adecuada para las redes en anillo, y la CSMA/CD, que se usa en la mayoría de las redes en bus, tanto si es un *bus* cableado como a través de un concentrador.

En la tercera parte, se han descrito de los protocolos que se utilizan en la red Internet. En concreto, se han visto los protocolos del nivel de interconexión de red, que son IP y sus asociados ARP e ICMP, y los protocolos del nivel de transporte: TCP y UDP.

Se ha visto el fenómeno del encapsulamiento de la información, como resultado de tener diferentes protocolos involucrados en una misma conexión, y como este encapsulamiento afecta al nivel de red.

Uno de los aspectos más relevantes del protocolo IP es la asignación de direcciones. Hemos visto como cada interfaz conectada a Internet debe tener una dirección IP única que la identifique. Las direcciones IP no sólo identifican estaciones, sino también la red o subred donde está la estación. De este modo, es posible encaminar los paquetes IP a través de diferentes encaminadores y, por lo tanto, a través de diferentes redes.

El protocolo ICMP nos permite resolver incidencias que puedan ocurrir en la Red. Hemos descrito las peculiaridades de este protocolo, así como dos utilidades como son el `ping` y el `traceroute`, de uso muy habitual.

Hemos descrito las redes de acceso a Internet más habituales, como son la red Ethernet, el acceso mediante una conexión telefónica con el protocolo PPP y el acceso mediante el protocolo ADSL, que permite usar el cable telefónico pero sin tener que establecer una llamada.

ANOTACIONES

Por lo que respecta al nivel de transporte, hemos visto como su principal objetivo es entregar la información a los niveles orientados a la aplicación en los extremos de la red. En la jerarquía de protocolos TCP/IP se definen dos protocolos de transporte:

1) El UDP, que es un protocolo no orientado a la conexión. Por tanto, no efectúa ningún control de errores ni de flujo. Si un datagrama UDP llega equivocado (el UDP utiliza un código detector de errores), el UDP lo descarta y no lo entrega a la aplicación. Esta última deberá ser capaz de responder a este tipo de servicio o deberá asumir la pérdida de la información. Este tipo de servicio puede ser útil en aplicaciones en tiempo real, en las que es más importante que la información llegue cuando le toca; es decir, con un retardo limitado, que no que se pierda una parte de la misma.

2) El TCP, que es un protocolo orientado a la conexión. Habrá una fase de establecimiento de la conexión (el llamado procedimiento *three-way handshake*), una fase de transmisión de la información y una fase de finalización de la conexión. El TCP entregará la información a la aplicación totalmente libre de errores. Para conseguirlo, necesita efectuar un control de errores y de flujo. El TCP utiliza un código detector de errores junto con un protocolo de retransmisiones para recuperar la información errónea. Como las memorias intermedias de recepción se pueden desbordar, el TCP utiliza un control de flujo por ventana deslizante. El TCP debe dimensionar correctamente los temporizadores de retransmisión.

En la cuarta y última parte, se ha presentado el concepto de programación distribuida y el modelo cliente/servidor, que es el más utilizado. También se ha presentado el modelo *peer-to-peer*, que no es una alternativa al anterior como a veces se ha planteado, porque no es un modelo de programación de aplicaciones, sinó que hace referencia al tipo de máquinas que se interconectan.

También se ha descrito el servicio de nombres de dominio, que proporciona acceso a una base de datos distribuida por toda la Red que permite obtener información sobre un determinado nombre. Las consultas más habituales son para averiguar la dirección IP que corresponde a un nombre de servidor, pero también es posible obtener el nombre del servidor de correo de un dominio dado, el nombre de una máquina a partir de su dirección IP, etc.

ANOTACIONES

Por lo que respecta a las propias aplicaciones, en esta unidad se han presentado los llamados servicios básicos de Internet, que son:

- Telnet: protocolo general para establecer sesiones interactivas con un servidor que permite que el cliente actúe como un terminal virtual.

- rlogin: protocolo de terminal virtual con facilidades para simplificar el proceso de identificación del usuario.

- rsh, rexec: protocolos para la ejecución remota de procesos en el servidor.

Se han presentado también los protocolos más usados quizás hoy en día, como son la transferencia de archivos, el correo electrónico, el servicio de *news*, el WWW y la mensajería electrónica, con sus múltiples posibilidades.

Sobre el FTP, se ha comentado que permite copiar archivos del cliente al servidor, del servidor al cliente y también de un servidor a otro, y que proporciona operaciones para que el cliente pueda manipular el sistema de archivos del servidor.

Sobre el correo electrónico, que es un servicio basado en los mismos conceptos del correo postal, se ha comentado en concreto la filosofía de almacenamiento y reenvío.

La norma RFC 822 especifica el formato de los mensajes, y la norma MIME especifica un conjunto de extensiones para, por ejemplo, adjuntar archivos y usar diversos idiomas.

Tres protocolos proporcionan las diferentes funcionalidades necesarias:

- SMTP: transferencia de mensajes.

- POP3: acceso simple a buzones de correo.

- IMAP4rev1: acceso complejo a buzones de correo.

ANOTACIONES

El servicio de noticias (*news*) facilita la publicación de mensajes con un alcance que puede ir desde el ámbito local de una organización, hasta el ámbito mundial (toda la Internet). Se basa en la propagación de los mensajes entre los diferentes servidores por medio del NNTP. Este mismo protocolo puede ser utilizado por los clientes para acceder a los mensajes guardados en un servidor o para enviar otros nuevos.

El servicio WWW permite "navegar'' por un conjunto de elementos de información interconectados con referencias o enlaces entre sí que pueden contener datos con diferentes tipos de representaciones (texto, imágenes, audio, vídeo). El protocolo que proporciona el acceso a esta información es el HTTP.

El servicio de mensajería instantánea está soportado por diferentes protocolos, algunos propietarios. Existen diferentes plataformas que lo proporcionan y también se han desarrollado diferentes clientes multiprotocolo que permiten acceder a diferentes redes de mensajería simultáneamente.

ANOTACIONES

Bibliografía

Comer, D.E. (2000). "Principles, Protocols and Architecture". *Internetworking with TCP/IP* (vol. 1). (4.ª ed.). Upper Saddle River: Prentice Hall.

Pierce, J.R.; Noll, A.M. (1995). *Señales. La ciencia de las telecomunicaciones.* Barcelona: Reverté.

Rose, Marshall T. (1990). *The open book.* Englewood Cliffs: Prentice Hall.

Stallings, W. (2000). *Comunicaciones y redes de computadores* (6.ª ed.). Madrid: Prentice-Hall.

Stevens, W.R. (1994). "The Protocols" *TCP/IP Illustrated* (vol. I). Wilmington: Addison-Wesley.

Tanenbaum, A.S. (2003). *Redes de computadoras* (4.ª ed.). Méjico: Pearson Educación.

ANOTACIONES

Anexos

Anexo 1

El algoritmo *checksum*

Los protocolos IP y TCP utilizan, entre otros, un sencillo *checksum* para la detección de errores. Ésta es una versión del algoritmo *checksum*:

```
u_short checksum(u_short * addr,int len)
{
   int aux_len = len;
   u_short *aux_addr = addr;
   u_short res;
   int sum = 0;

   while(aux_len > 1)
   {
   sum+ = *aux_addr++;
   aux_len- = 2;
   }

   if(aux_len == 1)
   sum+ = *(u_char) *aux_addr;

   sum = (sum>>16) + (sum & 0xffff);
   sum+ = (sum>>16);
   res = ~sum;

   return res;
}
```

ANOTACIONES

Anexo 2

Aplicaciones mencionadas en el texto

1) Aplicaciones estándar

A continuación, resumimos las aplicaciones estándar que se han visto en el cursoy que están disponibles en entornos Unix y GNU/Linux.

Nota

En los sistemas Unix podréis encontrar más información utilizando el comando man. Por ejemplo, para saber más detalles del comando route de Unix podéis ejecutar:

```
$ man route
```

- **`ping`:** por medio de paquetes ICMP (del tipo petición de eco y respuesta de eco) permite saber si una máquina está funcionando o no y tener una idea de cuál es el retardo de ida y vuelta (*round-trip*). Asimismo, permite saber por qué máquinas pasa el paquete hasta que llega a destino. Para esta función, va mejor el programa `traceroute`, a causa de las limitaciones inherentes a los paquetes ICMP.

Nota

Para ejecutar los comandos que se muestran en este anexo se requiere tener privilegios de superusuario.

- **`traceroute`:** permite averiguar la ruta que se sigue desde el equipo local hasta un destino cualquiera de Internet. Marca los retardos existentes en cada uno de los nodos que es preciso cruzar. Se basa en el campo TTL del paquete IP y los mensajes ICMP-*time-to-live-exceeded*. Esta aplicación no está disponible en algunas distribuciones de Unix y distribuciones de GNU/Linux, pero se puede conseguir facilmente.

- **`arp`:** permite consultar y modificar la tabla ARP (caché ARP) de una máquina.

- **`route`:** sirve para consultar y modificar la tabla de direccionamiento IP de una máquina conectada a Internet.

- **`ifconfig`:** permite consultar el estado de las tarjetas de red disponibles en el sistema local.

- **`netstat`:** proporciona estadísticas de los protocolos TCP y UDP. Permite listar los puertos que se escuchan. Muestra el estado en que se encuentran los *sockets* TCP. Si queréis un listado de todos los *sockets* activos y puertos abiertos, haced `netstat -a`, y, si os interesan otras variantes, consultad la ayuda.

- **`telnet`:** ofrece, aparte de la principal emulación de terminal, conectar y experimentar los protocolos ASCII.

ANOTACIONES

2) Aplicaciones no estándar

- **netcat** (licencia GPL-Lenguage C): esta aplicación nos permite utilizar conexiones TCP y UDP desde la línea de comandos (por ejemplo, transmitir un fichero) o comprobar qué puertos tiene abiertos una determinada máquina, entre otros servicios.

- **tcpdump** [Unix (software libre-lenguaje C)]: permite analizar el tráfico de la red por medio de conexión (LAN o punto a punto). Al contrario de lo que su nombre indica, es capaz de interpretar los paquetes no sólo en el ámbito TCP, sino también en el IP, red, y aplicación (para aplicaciones comunes). Es una herramienta imprescindible para cualquier administrador de sistemas, aunque no aparezca en las distribuciones de las diferentes variantes de Unix. El código es de libre distribución. Sus autores son Van Jacobson, Craig Leres y Steven McCanne, de la Universidad de California, aunque el programa en que se basaba el original, el Etherfind, era propiedad de Sun Microsystems y se distribuía dentro de SunOS.

Nota

En LAN, el `tcpdump` pone la tarjeta de red en modo promiscuo. Lo que significa que todo el tráfico de la red es visible, con lo que cualquier usuario malintencionado (un *hacker*) puede hacer un mal uso del mismo. Por tanto, el tcpdump y otras herramientas similares o derivadas del `tcpdump`, conocidas como detectores (*sniffers*) se pueden considerar como una herramienta potencialmente peligrosa. De hecho, lo que es arriesgado es no utilizar al menos uno de los tres mecanismos de protección más importantes contra los detectores: cifrado, cifrado y cifrado.

A continuación, se indican los resultados que puede mostrar el programa **tcpdump**.

3) El programa **tcpdump**

El programa `tcpdump` permite capturar y filtrar paquetes que cruzan una interfaz de red que se ha activado en *modo promiscuo* (todos los

ANOTACIONES

paquetes que pasan por el medio son capturados y filtrados). Los paquetes son procesados por un software especial que sólo puede ejecutar el superusuario (*root*) de la máquina.

Para ver cómo funcionan los diferentes protocolos, proporcionamos ejemplos de las trazas que muestra el `tcpdump` durante el establecimiento o la terminación de la conexión.

Para ver cómo funciona este comando, puede utilizarse la instrucción `man` de Unix (o GNU/Linux). En este anexo ofrecemos un ejemplo del significado de la traza del `tcpdump`.

El formato general de salida del `tcpdump` es el siguiente:

```
Data  src> dst: flag data-sqno ack window urgent options
```

El significado de los componentes del formato son los siguientes:

- **`data:`** este componente nos proporciona la hora en que se produjo el acontecimiento en formato hora:minutos:segundos.microsegundos (o milisegundos, según la resolución del reloj).
- **`src`** y **`dst`**: son las direcciones IP y los puertos TCP/UDP de las conexiones de fuente y de destino.
- **`flags:`** son una combinación de los posibles indicadores de un segmento/datagrama TCP/UDP: S (SYN), F (FIN), P (PUSH), R (RST) y '.' (que significa que no hay indicadores).
- **`data-sqno:`** describe el número de secuencia de la porción de datos.
- **`ack:`** es el número de secuencia del byte siguiente que espera recibir el otro extremo TCP/UDP.
- **`window:`** es el tamaño de la ventana que el receptor advierte al transmisor.
- **`urgent:`** indica que existen datos urgentes en este segmento/datagrama.
- **`options:`** son las opciones TCP que suelen estar entre corchetes del tipo < >; por ejemplo, el tamaño máximo del segmento (por ejemplo, <mss 1.024>).

ANOTACIONES

Supongamos que hacemos un `rlogin` desde la máquina *argos* (fuente) hasta la máquina *helios* (destino). Un ejemplo de una línea que retorna el comando `tcpdump` sería el siguiente:

- `12:34:23.165439 argos.1023 > helios.login: S 7838437: 7838437 (0) win 4096 <mss 1024>`
- `12:34:23.165439 helios.login > argos.1023: S 453534: 453534 (0) ack7838438 win 4096 <mss 1024>`

La primera línea indica lo siguiente:

a) El acontecimiento tiene lugar a las 12:34:23.165439.

b) Su origen es la máquina *argos* con el puerto 1.024. Su destino es la máquina *helios* con el puerto *login*.

c) El indicador S señala que es un segmento de SYN (por ejemplo, por un inicio de conexión).

d) Advierte un número de secuencia 7.838.437 y consume hasta este número. Conviene observar que el `tcpdump` escribe el número inicial y el final del número de secuencia de datos, y, entre paréntesis, la diferencia indicando la longitud del campo de datos (en este caso 0 bytes, puesto que es un segmento de petición de conexión). Por tanto, indica entre paréntesis la longitud del segmento de datos. Este número de secuencia es absoluto. Las salidas siguientes del `tcp-dump` indicarán los números de secuecia relativos al inicial. Por ejemplo, en lugar de indicar 7.838.437: 7.838.450, (13) en notación absoluta, indicará 1:13, (13) en notación relativa al ISN (se cumple que valor absoluto es igual al ISN más el valor relativo). Lo mismo se aplica a los números de secuencia del ACK.

e) Advierte una ventana de 4.096 bytes.

f) Hay una petición de tamaño máximo de segmento de 1.024 bytes.

La segunda línea indica lo siguiente:

a) El origen es la máquina *helios* con el puerto `login`. El destino es la máquina *argos* con el puerto 1.023.

b) El indicador *S* señala que es un segmento de SYN (por ejemplo, para un inicio de conexión).

c) Indica el número de secuencia inicial 453.534.

d) Reconoce con un ACK el byte siguiente que espera del transmisor, el 7.838.438 (es decir, 7.838.437 + 1), puesto que los recibió correctamente hasta el 7.838.437).

e) Advierte una ventana de 4.096 bytes.

f) Hay una petición de tamaño máximo de segmento de 1.024 bytes.

El programa `tcpdump` permite escuchar, de manera sencilla, todo lo que sucede en la red. Admite toda una serie de indicadores para filtrar sólo las direcciones IP de fuente o de destino que pueden interesar, o el tipo de protocolo que se quiere escuchar (TCP, UDP, ARP, etc.). Asimismo, admite indicadores para obtener el campo de datos, para filtrar un número fijo de segmentos, etc. Para ver todas las posibles opciones que permite el programa `tcpdump` se utiliza el comando `man`.

Anexo 3

Notación

En este anexo se detalla la notación que se utiliza en toda la unidad para describir los campos de los mensajes y los comandos soportados por los diferentes protocolos.

Cadena literal

Las cadenas literales se definirán con letra de espaciado fijo.

```
cadena
```

Variable

Las variables se definirán con letra inclinada de espaciado fijo o letra cursiva.

```
variable
```

Alternativa

Para indicar que en una especificación se debe elegir un elemento de una lista de *n* elementos, se utiliza el carácter "|" para separar las diferentes alternativas.

```
elemento1 | elemento2 | ... | elementon
```

Repetición

Para los elementos que se pueden repetir entre *n* y *m* veces, se utiliza la construcción *n*m* como prefijo. Los valores *n* y *m* se pueden suprimir. En este caso, se toman por defecto los valores $n = 0$ y $m =$ vacío.

```
n*melemento
n*elemento
*melemento
*elemento
```

ANOTACIONES

Opcional

Para indicar que una cadena o una variable son opcionales, es preciso cerrarlas entre los caracteres "`[`" y "`]`".

```
[elemento]
```

Repetición específica

Cuando se quiere indicar que un elemento debe aparecer con exactitud un número entero *n* de veces, debe ponerse el número *n* como prefijo.

```
nelemento
```

Lista

Si se quiere especificar una lista de entre *n* y *m* elementos separados por comas, se utiliza la construcción `n#m` como prefijo. Los valores *n* y *m* se pueden suprimir. En este caso se toman los valores por defecto, que son $n = 0$ y $m =$ vacío.

```
n#melemento
```

Agrupamiento

Cuando se quiere que uno o más elementos sean tratados como un único elemento a efectos de los operadores `|`, `*` y `#`, se indica cerrándolos con los caracteres "`(`" y "`)`". Se suele utilizar en especificaciones de repeticiones y listas.

```
(elemento1 elemento2 ... elementon)
```

Caracteres especiales

Algunos caracteres especiales se incluyen entre los caracteres "<" y ">".

```
<CR>: retorno de carro (CR, carriage return)
<LF>: salto de línea (LF, line feed)
<CRLF>: retorno de carro + salto de línea
```

Anexo 4

Códigos de respuesta

Las diferentes RFC definen los códigos de respuesta posibles para cada uno de sus comandos, así como su significado. Estos códigos están formados por tres dígitos y, para facilitar su interpretación, llevan asociado un significado concreto para cada dígito.

Primer dígito

Indica si la operación se ha efectuado o no.

Tabla 18.

Primer dígito	
1yz	Respuesta preliminar: se ha iniciado la operación y, cuando se haya completado, el servidor enviará una nueva respuesta.
2yz	Operación completada correctamente.
3yz	Respuesta intermedia: la operación se ha aceptado; sin embargo, el cliente debe enviar nuevos comandos con más información para completarla.
4yz	Respuesta negativa temporal: no se puede llevar a cabo la operación, pero se podrá efectuar si el cliente lo reintenta.
5yz	Respuesta negativa definitiva: no se puede efectuar la operación y probablemente tampoco se podrá llevar a cabo si el cliente lo reintenta.

Nota

Cuando se quiere leer un fichero con acceso temporalmente bloqueado porque otro proceso está escribiendo datos en el mismo, el servidor envía una respuesta negativa temporal. En cambio, cuando se quiere leer un fichero que no existe, el servidor envía una respuesta negativa definitiva, aunque siempre existe la posibilidad de que mientras tanto otro proceso lo cree.

Segundo dígito

Indica el tipo de respuesta.

Tabla 19.

Segundo dígito	
x0z	Error de sintaxis, comando no implementado o, en general, respuesta no perteneciente a ninguna de las otras categorías.
x1z	Respuesta informativa.
x2z	Respuesta referente a las conexiones.
x3z	Respuesta referente al proceso de autenticación (nombre de usuario, contraseña, etc.).
x5z	Respuesta referente a la aplicación concreta (por ejemplo, al sistema de ficheros o de correo).

ANOTACIONES

Tercer dígito

El tercer dígito complementa la información del segundo dígito para especificar de qué respuesta concreta se trata.

GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

ANOTACIONES

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

ANOTACIONES

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

ANOTACIONES

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

ANOTACIONES

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material.

If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

ANOTACIONES

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

ANOTACIONES

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

ANOTACIONES

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number.

Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

ANOTACIONES

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form.

Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the

ANOTACIONES

translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

ANOTACIONES

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

ANOTACIONES